



UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Civil, Arquitetura E Urbanismo

VICTOR CALIXTO

**GERAÇÃO AUTOMATIZADA DE *LAYOUTS* COM USO
DE ALGORITMOS EVOLUTIVOS:
Aplicações em Arquitetura e Urbanismo**

Campinas

2016

VICTOR CALIXTO

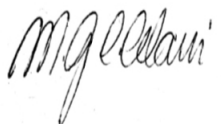
**GERAÇÃO AUTOMATIZADA DE *LAYOUTS* COM USO
DE ALGORITMOS EVOLUTIVOS:
Aplicações em Arquitetura e Urbanismo**

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 : Prof^a. Dr^a. Maria Gabriela Caffarena Celani

ESTE EXEMPLAR CORRESPONDE À
VERSÃO FINAL DA DISSERTAÇÃO DEFEN-
DIDA PELO ALUNO VICTOR CALIXTO,
E ORIENTADA PELA PROF^a.DR^a. MARIA
GABRIELA CAFFARENA CELANI

ASSINATURA DA ORIENTADORA:



Campinas

2016

Agência(s) de fomento e nº(s) de processo(s): CAPES, 01-P-04375-2015

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

C29g Calixto, Victor, 1987-
Geração automatizada de layouts com o uso de algoritmos evolutivos : aplicações em arquitetura e urbanismo / Victor Calixto. – Campinas, SP : [s.n.], 2016.

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

1. Computação evolutiva. 2. Layout. 3. Algoritmos. 4. Arquitetura - Urbanismo. 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: Automated space planning using evolutionary algorithms : applications in architecture and urbanism

Palavras-chave em inglês:

Evolutionary computation

Layout

Algorithms

Architecture - Urbanism

Área de concentração: Arquitetura, Tecnologia e Cidade

Titulação: Mestre em Arquitetura, Tecnologia e Cidade

Banca examinadora:

Maria Gabriela Caffarena Celani [Orientador]

Hélio Pedrini

Juarez Moara Santos Franco

Data de defesa: 05-12-2016

Programa de Pós-Graduação: Arquitetura, Tecnologia e Cidade

**GERAÇÃO AUTOMATIZADA DE *LAYOUTS* COM O USO DE
ALGORITMOS EVOLUTIVOS :**

Aplicações em Arquitetura e Urbanismo

Victor Calixto

Dissertação de mestrado aprovada pela Banca Examinadora, constituída por:

Profa.Dra. Maria Gabriela Caffarena Celani

Presidente e Orientadora – Universidade Estadual de Campinas UNICAMP

Prof. Dr. Juarez Moara Santos Franco

Faculdade de Engenharia Civil, Arquitetura e Urbanismo – UNICAMP

Prof. Dr. Hélio Pedrini

Instituto de Computação – Unicamp

A Ata da defesa com as respectivas assinaturas dos membros encontra-se no processo de vida acadêmico do aluno.

Campinas, 05 dezembro de 2016

*In memoriam de Maria Aparecida, minha avó, por todos os anos de carinho e dedicação
aos filhos e netos.*

Agradecimentos

À orientação da professora **Gabriela Celani**, pela amizade e pelo exemplo de pessoa, professora e profissional, sempre buscando incentivar seus orientandos a pensarem além a cada dia.

Aos **professores do pós ATC**, pela dedicação ao programa e por buscarem explorar o potencial dos alunos.

À **pós ATC** pela oportunidade e à **CAPES** pelo apoio financeiro no desenvolvimento dessa pesquisa.

Aos professores da **banca examinadora** por gentilmente cederem um pouco do seu tempo para contribuir com esta dissertação.

Aos alunos e pesquisadores que participaram e colaboraram com a realização do Workshop de Space Planning : **Rômulo Avi Oliveira, Guilherme Giantini, Vinicius Mizobuti, Fernanda Destro Tonelli, Rogério Lima, Verley Henry Coco Jr, Aderson passos, Gabriel Ribeiro, Dyego Digiandomenico, Felipe Tricoli, Adriano Lima, Rodrigo Makert, Ronaldo A. Marques Pinto e Vinicius Gonçalves.**

Aos pesquisadores e amigos do LAPAC/UNICAMP, que contribuíram com discussões enriquecedoras: **Filipe Campos, André Araújo, Daniel Lenz, Wilson Barbosa, Letícia Mendes, Maycon Sedrez, Jarryer Martino, Natália Nakamura, Márcia Anaf Wagner e Lorena Moreira.**

Aos **funcionários do LAPAC** e da **pós ATC**.

Aos amigos de trabalho, workshops e pesquisas: **Affonso Orciuoli, Maurício Matias, Renata Portelada, Thomas Yulki Takeuchi, Orion Campos, Victor Sardenberg, Charles Vincent, Tony Guedes, Ernesto Bueno, Alex Garcia, Andrés Passaro, Gonçalo Henriques, Clarice Rohde, Carina Carmo e Bianca Ilha.**

À **Juliana Vianna Croffi** pelo companheirismo e amor.

Aos meus pais, **Benedito Calixto** e **Rosana Aparecida dos Santos**, e minhas irmãs **Michele Aparecida dos Santos Calixto** e **Thamiris Calixto** por colaborarem com minha formação, e por estarem me acompanhando e me apoiando em minhas decisões.

E por fim, agradeço a todas as pessoas que colaboraram com a minha formação pessoal, profissional e acadêmica, e que direta ou indiretamente colaboraram para a conclusão deste trabalho.

"Our technology, our machines, is part of our humanity. We created them to extend ourselves, and that is what is unique about human beings."
(Ray Kurzweil)

Resumo

As novas ferramentas computacionais têm modificado significativamente o discurso arquitetônico nas últimas décadas. O projeto computacional teve sua origem no início dos anos 60, e uma das metodologias que foi amplamente discutidas na época foi o *Space Planning* (SP), um campo de pesquisa que estuda o processo de arranjo de uma série de elementos topológicos e/ou geométricos, em um determinado espaço, em que distância, adjacência e outras funções de arranjo são o principal objetivo. Esta pesquisa buscou entender e explorar as possibilidades dos princípios da resolução de problemas de *space planning* relacionados à geração automatizada de *layouts* em arquitetura e urbanismo, otimizados através de algoritmos evolutivos. A abordagem do trabalho foi teórico-prática, a implementação dos algoritmos foi feita de maneira sistemática e com complexidade incremental, testando a viabilidade da aplicação através de um workshop realizado na UNICAMP. O estudo permitiu concluir que, por meio da utilização de um ambiente de programação visual combinado com linguagem textual dentro de um ambiente CAD (*Computer Aided-Design*), o método se mostra bastante flexível e relativamente fácil de usar, mesmo por arquitetos sem experiência anterior neste tipo de sistema. Embora sejam ainda necessários estudos mais aprofundados para o desenvolvimento de uma interface amigável, a viabilidade de emprego do método se mostra bastante promissora.

Keywords: *Space Planning*; Geração automatizada de *layouts*; Computação Evolutiva; Arquitetura Generativa.

Abstract

The new computational tools have significantly changed the architectural discourse in recent decades. However, the computational design originated in the early 60s, and one of the methodologies that were widely discussed at the time was the Space Planning (SP), which is a research field that studies the arrangement process of a number of topological elements and / or geometric, in a given space, where distance, adjacency and other arrangement functions are the main goal. This research sought to understand and explore the possibilities of the principles of resolution planning space issues related to automated generation of layouts in architecture and urbanism, optimized through evolutionary algorithms. The work approach was theoretical and practical, the implementation of the algorithms was done systematically and with incremental complexity, testing the viability of the method through a workshop held at Unicamp. The study concluded that, by using a combined visual programming environment with textual language in a CAD environment, the method proves very flexible and relatively easy to use, even by architects with no previous experience in this type of system. Although still needed further studies for the development of a user friendly interface, the feasibility of using the method shown to be very promising.

Keywords: Space Planning; Evolutionary Computation; Generative Architecture.

Lista de Ilustrações

Figura 1 – BMW-WELT-Coop Himmelblau	21
Figura 2 – Federation Square - Lab Architecture Studio	22
Figura 3 – Pavilhão realizado em uma pesquisa na Universidade de Stuttgart-ICD-ITKE 2014	22
Figura 4 – Pavilhão realizado em uma pesquisa na Architectural Association [C]Pavilion AADRL 2007	23
Figura 5 – Consultoria de Design to production - Mercedes Benz Museum	24
Figura 6 – Consultoria de Programming architecture - Shenzhen airport	24
Figura 7 – Diferentes configurações de plantas residências projetadas por Wright com uma mesma relação de topológica entre os ambientes	38
Figura 8 – Soluções de projeto de acordo com o objetivo	39
Figura 9 – Relação entre tempo computacional e dados de entrada	40
Figura 10 – Distribuição em árvore dos construction algorithms	41
Figura 11 – Grafo (V,A)	44
Figura 12 – Grafo planar(A) e Grafo não planar(B)	45
Figura 13 – Grafo planar máximo	45
Figura 14 – Matriz de adjacência de Miller	45
Figura 15 – Matriz de adjacência revisada para maximizar diagonal de Miller	46
Figura 16 – Matriz de adjacência revisada mapeada em um grafo	46
Figura 17 – Exemplo de grafo dual	46
Figura 18 – Grafo dual de Grason	47
Figura 19 – Diferentes configurações espaciais dos grafos à esquerda e subgrafos a direita	48
Figura 20 – Diferentes soluções de layout para um mesmo problema GRAMPA	48
Figura 21 – Grid com uma serie de números se movendo baseados em nas regras de Barricelli	53
Figura 22 – Fluxo de funcionamento dos algoritmos evolutivos	55
Figura 23 – Representação do DNA	56
Figura 24 – Representação de uma casa através dos fenótipos e genótipos	57
Figura 25 – Espaço de busca e espaço solução	58
Figura 26 – Fluxograma do funcionamento de um algoritmo genético.	60
Figura 27 – Operador de recombinação	61
Figura 28 – Operador de mutação	61
Figura 29 – Seleção de roleta	62
Figura 30 – Recombinação através de 3 pontos de corte	65
Figura 31 – Fluxograma do funcionamento das estratégias evolutivas	67
Figura 32 – Fluxograma do funcionamento da programação evolutiva	68

Figura 33 – Representação hierárquica de um cromossomo	69
Figura 34 – Recombinação PG	70
Figura 35 – Funções e Procedimentos de uma PG através de linguagem LISP e expressão polonesa	71
Figura 36 – Projeto evolutivo como consequência da interseção de distintas áreas	72
Figura 37 – Espaçadores, cobertura e experimentos de Jarryer Martino	75
Figura 38 – População de cobertura através de gerações	75
Figura 39 – Gráfico das áreas de sombreamento em m^2 vs gráfico do somatório dos vãos em metros.	76
Figura 40 – Valores de <i>fitness</i> dos experimentos	77
Figura 41 – Área de sombreamento Vs comprimento dos vãos	78
Figura 42 – População gerada através do método aditivo	79
Figura 43 – População gerada através do método subtrativo	80
Figura 44 – Resultados obtidos através dos experimentos aditivos novos e método de cheios e vazios	81
Figura 45 – Fluxo de dados de um processo determinístico e de um processo estocástico	83
Figura 46 – Representação do fatiamento de um retângulo e sua representação através de uma <i>slicing tree</i>	88
Figura 47 – Diferentes operadores representando diferentes maneiras de definição dos <i>layouts</i>	89
Figura 48 – Comparação feita entre o método de <i>hill climbing</i> e o método de AG.	90
Figura 49 – Comparações feitas por Perez, Shoenefeld e Wainwright entre métodos de STS.	91
Figura 50 – Layout produzido por Perez, Shoenefeld e Wainwright para um edifício de 20 ambientes.	92
Figura 51 – Representação do fenótipo <i>bottom-up</i> da solução (D(D 2 (R(0 1)(R 3 4))). .	92
Figura 52 – Otimização de soma ponderada e método ranqueado	93
Figura 53 – Sistema LHS - RHS	95
Figura 54 – Sistemas de conexão através de marcadores	96
Figura 55 – Processo de conexão entre ambientes	96
Figura 56 – Fluxograma da codificação dos indivíduos realizado por Jo e Gero	97
Figura 57 – Edifício de serviços de 4 andares desenvolvido por Ligget em 1985	98
Figura 58 – (a) Solução produzida pelo método Ligget e (b) Solução produzida pelo método EDGE após 500 gerações	99
Figura 59 – Representação dos ambientes por caracteres	100
Figura 60 – Custo de alocação entre ambiente	100
Figura 61 – Custo de transporte entre ambiente	101
Figura 62 – Resultados obtidos com programação genética por Jagielski e Gero	101
Figura 63 – Resumo dos resultados dos experimentos de Li e Love	102

Figura 64 – Fluxograma do algoritmo desenvolvido por Garza e Maher	103
Figura 65 – Fluxograma do algoritmo desenvolvido por Garza e Maher	104
Figura 66 – Fluxograma de funcionamento do algoritmo ESAP	105
Figura 67 – Resultado de um experimento multi-pavimento realizado por Rodrigues, Gaspar e Gomes	106
Figura 68 – Vetores de orientação constituindo ambientes	107
Figura 69 – População de ambientes gerados por Rosemman e Gero	107
Figura 70 – População de zonas geradas por Rosemman e Gero	108
Figura 71 – População de edificações geradas por Rosemman e Gero	108
Figura 72 – Interface do programa desenvolvido por Elezkurtaj e Franck	109
Figura 73 – Método de blocking: (a) genótipo (b) fenótipo	109
Figura 74 – Resultado do experimento 4	110
Figura 75 – Ilustração do funcionamento do processo de colaboração desenvolvido por Banerjee, Quiroz e Louis	111
Figura 76 – Operação recombinação entre dois indivíduos da população	111
Figura 77 – <i>Layout</i> de 1 a 6 avaliados de forma não colaborativa e de 7 a 12 de maneira colaborativa	112
Figura 78 – Valores obtidos na avaliação dos <i>layouts</i>	112
Figura 79 – Formas livres desenhadas pelo usuário	113
Figura 80 – Adaptação das formas livres ao grid	113
Figura 81 – Evolução das forma através do AG	113
Figura 82 – Espaço de 0 - 12 tendiam a se aglomerar formando o que ocasionou em sua identificação como gene evoluído	114
Figura 83 – Comparação entre a convergência da solução entre AG e AG com EG . . .	115
Figura 84 – Sétima geração de um experimento realizado por Houmayouni	116
Figura 85 – Matriz de adjacência das relações entre os ambientes do algoritmo realizado por Houmayouni	116
Figura 86 – Interações realizadas na escala do edifício do algoritmo de Houmayouni . . .	117
Figura 87 – Resultados dos layouts obtidos através de um experimento realizado por Houmayouni	117
Figura 88 – Processo de recombinação através da matriz de adjacência	118
Figura 89 – <i>Adjacency Preference Scale</i> (APS) ranqueamento dos ambientes	119
Figura 90 – (a) espaço (b) <i>half plan</i> que define o espaço	120
Figura 91 – Processo evolutivo dos Half Planes realizado por Dasmki e Gero	121
Figura 92 – Processo evolutivo dos <i>layouts</i> realizado por Dasmki e Gero	121
Figura 93 – Processo de decomposição do espaço através de KD-trees	122
Figura 94 – Interface do processo de otimização definido por Knecht	122
Figura 95 – Número de artigos publicados por ano	123
Figura 96 – Métodos evolutivos e complementares identificados nos artigos selecionados	124

Figura 97 – Métodos de concepção do <i>layout</i> identificados nos artigos selecionados. . . .	124
Figura 98 – Métodos de representação do cromossomo identificados nos artigos selecionados	125
Figura 99 – Métodos de avaliação utilizados nos artigos selecionados	125
Figura 100 – Número máximo de ambientes computados nos artigos analisados, por ano .	126
Figura 101 – Número máximo de gerações nos artigos analisados por ano e número máximo da população por ano respectivamente.	126
Figura 102 – Número máximo de gerações nos artigos analisados por ano e número máximo da população por ano respectivamente.	127
Figura 103 – Interface Rhinoceros 3D	130
Figura 104 – Interface Grasshopper	131
Figura 105 – Interface Galápagos	132
Figura 106 – Agrupamento simples	133
Figura 107 – Agrupamento simples com variáveis paramétricas	134
Figura 108 – Regra Colisão	135
Figura 109 – Regra Sobreposição	135
Figura 110 – Regra Adjacência	136
Figura 111 – Primeiro Experimento	137
Figura 112 – Componente Python "chamando" janela	140
Figura 113 – Módulos necessários para criação da janela	141
Figura 114 – Sequência de janelas do <i>plug-in Human UI</i>	142
Figura 115 – Automatização dos componentes do <i>Grasshopper</i> através de <i>script</i>	143
Figura 116 – Componente " <i>Enable/Disable Objects</i> " do <i>Metahopper</i>	143
Figura 117 – Terreno da implantação bidimensional	144
Figura 118 – Janela de configuração dos elementos	145
Figura 119 – Infográfico do fluxo de dados do algoritmo de implantação bidimensional .	146
Figura 120 – Gráfico das 50 gerações	146
Figura 121 – Distâncias entre elementos adjacentes e ponto de atração	147
Figura 122 – Resultado final do experimento	148
Figura 123 – Janela de configuração dos elementos <i>layout</i>	149
Figura 124 – Gráfico do fluxo de dados do algoritmo de <i>layouts</i>	150
Figura 125 – (a) Resultado do <i>layout</i> solução (b) Resultado do <i>layout</i> grafo	151
Figura 126 – Terreno da implantação com torres tridimensionais	152
Figura 127 – Dados de entrada implantação tridimensional	153
Figura 128 – Gráfico implantação tridimensional	154
Figura 129 – Resultado final experimento 3	154
Figura 130 – Visuais e Pontos de Interseção - (A) Visual de uma torre e um andar(B) Detalhe da visual (C) Visuais do conjunto de torres de todos os andares (D) Pontos de interseção das visuais de todos os andares com as demais torres e entorno	155

Figura 131 – Janelas de dados de entrada	156
Figura 132 – Gráfico da implantação tridimensional com visuais	157
Figura 133 – Resultado final da implantação tridimensional com visuais	157
Figura 134 – Implantação inicial dos módulos e penalidades	158
Figura 135 – Dados de entrada implantação tridimensional com blocos modulares	159
Figura 136 – Gráfico de implantação tridimensional com blocos modulares	160
Figura 137 – Resultado final da implantação tridimensional com blocos modulares	160
Figura 138 – Terreno do exercício proposto	164
Figura 139 – Imagem do Workshop	165
Figura 140 – Algoritmo de posicionamento do experimento do grupo 1 - Felipe Tricoli, Gabriel Ribeiro, Rômulo Avi Oliveira e Vinicius Mizobuti	166
Figura 141 – Algoritmo de simulação dos ventos e geometria produzida pelo grupo 1 - Felipe Tricoli, Gabriel Ribeiro, Rômulo Avi Oliveira e Vinicius Mizobuti	167
Figura 142 – Panorama geral do experimento do grupo 2 - Filipe Campos, Guilherme Giantini, Ronaldo A. Marques Pinto e Verley Henry Coco Jr	168
Figura 143 – Panorama geral do experimento do grupo 3 - Dyego Digiandomenico e Ro- gério Lima	169
Figura 144 – Panorama geral do experimento do grupo 4 - Adriano Lima, Rodrigo Makert e Vinicius Gonçalves	170

Lista de Tabelas

Tabela 1 – Representação e valores da taxa de proximidade	42
Tabela 2 – Tamanho fixo do cromossomo	59
Tabela 3 – Recombinação uniforme	65
Tabela 4 – Funções utilizadas no método	78
Tabela 5 – Relação de artigos analisados	85
Tabela 6 – Comparação feita por Kado entre diversos AG para space planning.	91
Tabela 7 – Comparação entre distintos métodos de seleção realizados por Flack	93
Tabela 8 – Representação e valores da taxa de proximidade	147
Tabela 9 – Distância entre os centros dos elementos e o ponto de atração	147
Tabela 10 – Ambientes e áreas desejadas e resultados	151
Tabela 11 – Representação e valores da taxa de proximidade	154
Tabela 12 – Representação e valores da taxa de proximidade	157
Tabela 13 – Representação e valores da taxa de proximidade	160

Lista de Abreviaturas e Siglas

AE	Algoritmos evolutivos
AEI	Algoritmo Evolutivo Interativo
AG	Algoritmo Genético
AGP	Algoritmo Genético Paralelo
ALDEP	<i>Automated Layout Design Program</i>
APS	<i>Adjacency Preference Scale</i>
CAAD	<i>Computer-Aided Architectural Design</i>
CAD	<i>Computer Aided Design</i>
CBR	<i>Case-Based Reasoning</i>
CRAFT	<i>Computerized Relative Allocation of Facilities Technique</i>
DNA	Ácido Desoxirribonucleico
EDGE	<i>Evolutionary Design using Genetic Evolution</i>
EE	Estratégias Evolutivas
EG	Engenharia Genética
ESAP	<i>Evolutionary Program for the Space Allocation Problem</i>
GRAMPA	<i>Graph Manipulating Package</i>
IA	Inteligência Artificial
LHS	<i>Left-Hand Side</i>
MEF	Máquina de estados finitos
MPC	<i>Minimum Closeness Preference</i>
PE	Programação Evolutiva
PG	Programação Genética
QAP	<i>Quadratic Assignment Problem</i>
RHS	<i>Right-Hand Side</i>

SA *Simulated Annealing*

SHC *Stochastic Hill Climbing*

SP *Space Planning*

STS *Slice Tree Structure*

Sumário

1	Introdução	21
1.1	O projeto computacional	21
1.2	Problemática e Justificativa	28
1.3	Objetivos	29
1.4	Relação com outras pesquisas	31
1.5	Metodologia	31
1.6	Estrutura da dissertação	33
I	Space Planning	35
2	<i>Space Planning</i>	36
2.1	Os Algoritmos de Space Planning	39
2.2	Algoritmos Heurísticos	40
2.2.1	<i>Construction Algorithms</i>	41
2.2.2	<i>Graph theoretic approach</i>	44
2.2.3	<i>Improvement Algorithms</i>	49
2.3	Considerações sobre <i>Space Planning</i>	50
II	Os Algoritmos Evolutivos	52
3	Algoritmos Evolutivos	53
3.1	Da codificação da vida à codificação de algoritmos	56
3.2	Os métodos evolutivos	58
3.2.1	Algoritmos Genéticos	59
3.2.1.1	Métodos de seleção	61
3.2.1.2	Os métodos de recombinação (<i>Crossover</i>)	64
3.2.1.3	Mutação	65
3.2.2	Estratégias Evolutivas	66
3.2.3	Programação Evolutiva	67
3.2.4	Programação Genética	68
3.2.4.1	Funções e Procedimentos	70
3.3	O projeto arquitetônico com abordagem evolutiva	71
3.3.1	Evolutionary Design by Computers – Bentley e os algoritmos evolutivos	72

3.3.1.1	Otimização do Projeto Evolutivo (<i>Evolutionary Design Optimization</i>)	72
3.3.1.2	Projeto Evolutivo Criativo (<i>creative evolutionary design</i>)	73
3.3.1.3	Algoritmo evolutivo com abordagem artística e formas artificiais de vida (<i>Evolutionary art / Evolutionary artificial life forms</i>)	73
3.4	O algoritmo evolutivo e algumas aplicações	74
3.4.1	Exploração criativa para o desenho de uma cobertura - Jarryer Martino 2015	74
3.4.2	Generative Programming and Spatial Morphogenesis- Paul Coates e Dimitrios Makris -(1999)	77
3.5	Considerações sobre os algoritmos evolutivos	81
III	Space Planning com abordagem evolutiva (1992-2014)	82
4	Space Planning com abordagem evolutiva	83
4.1	Revisão da Literatura 1992-2014	84
4.2	<i>Slice tree structure</i> (STS) com abordagem evolutiva	88
4.3	One-to-One Assignment com abordagem evolutiva	94
4.4	Abordagem interativa da função objetivo	107
4.5	Engenharia Genética no aperfeiçoamento das soluções	114
4.6	<i>Graph Theory</i> e o método evolutivo	118
4.7	<i>Half plan</i> e <i>Kd Trees</i> com abordagem evolutiva	120
4.8	Resultados da Revisão de literatura	123
4.8.1	Demandas para trabalhos futuros	127
4.8.2	Considerações sobre <i>space planning</i> com abordagem evolutiva	127
IV	Métodos e Materiais	129
5	Métodos e Materiais	130
5.1	Materiais	130
5.1.1	Especificações de Hardware e Software	130
5.2	Métodos	132
5.2.1	Procedimentos para execução dos experimentos aplicados	132
5.2.2	Método de implementação dos algoritmos	132
5.2.2.1	Agrupamento simples	132
V	Experimentos	138
6	Experimentos	139
6.0.1	Teste de Interface	140

6.0.2	Teste de Automatização de componentes	142
6.1	Experimentos aplicados	144
6.1.1	Experimento 1- Implantação bidimensional	144
6.1.2	Experimento 2 - Estudo de <i>Layout</i>	148
6.1.3	Experimentos 3 e 4 - Implantação com torres tridimensionais e im- plantação com torres tridimensionais com otimização das visuais . .	151
6.1.4	Experimento 5 - Implantação tridimensional com blocos modulares	157
6.2	Considerações finais sobre os experimentos	161
VI	<i>Workshop</i>	162
7	<i>Workshop</i>	163
7.1	Planejamento do Workshop	163
7.2	Workshop	165
7.3	Resultados	166
7.4	Discussão e Questionário	170
7.5	Considerações sobre o Workshop	171
VII	<i>Considerações Finais</i>	172
8	Considerações Finais	173
	Referências	175
	Apêndices	181
	APÊNDICE A Questionário	182
	APÊNDICE B Cronograma do Workshop	186
	APÊNDICE C Proposta do exercício do Workshop	188

1 Introdução

Nesse capítulo será feita uma introdução ao conceito de projeto computacional através de uma breve cronologia de eventos importantes, como também serão delineados a problemática e justificativa, os objetivos e a metodologia adotada no trabalho.

1.1 O projeto computacional

Nas últimas décadas, a utilização de ferramentas computacionais no processo e produção de design e arquitetura tem crescido de maneira significativa. Projeto paramétrico, ferramentas generativas e fabricação digital são metodologias que vêm sendo recorrentemente discutidas em escritórios de arquitetura e urbanismo (figuras 1 e 2), em cursos de graduação e pós-graduação de universidades e institutos internacionais (figuras 3 e 4), bem como em empresas especializadas em conectar indústrias e arquitetos (figuras 5 e 6).

Figura 1 – BMW-WELT-Coop Himmelblau



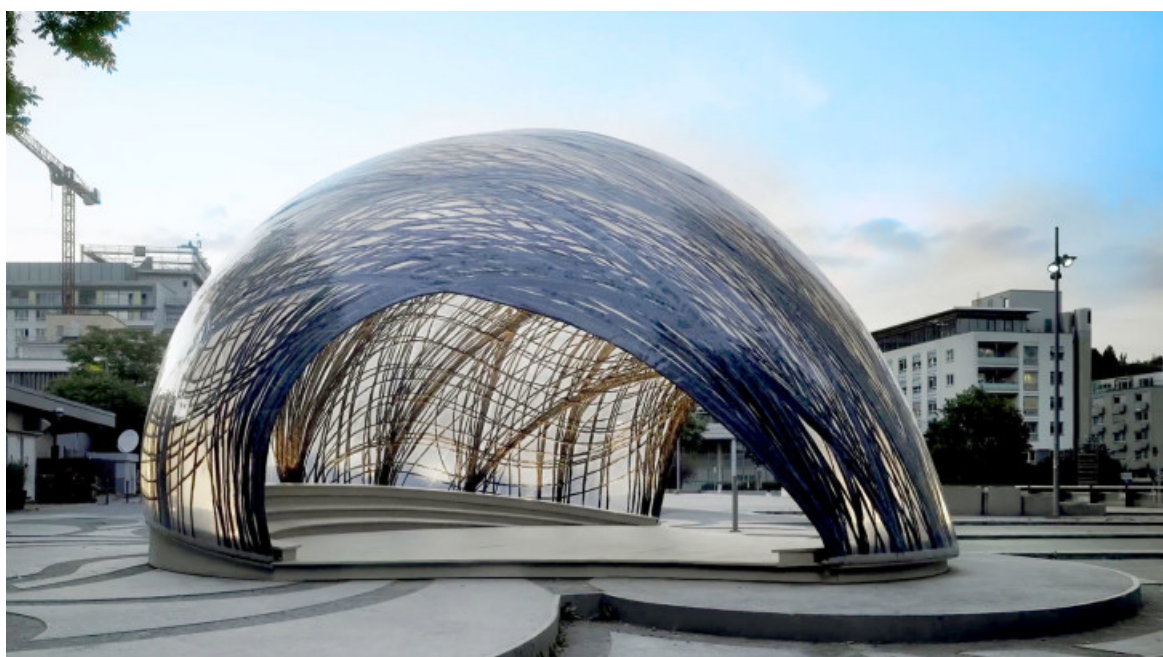
Fonte:upload.wikimedia.org - <https://upload.wikimedia.org/wikipedia/commons/5/54/M%C3%BCnchen_-_BMW-Welt.jpg>

Figura 2 – Federation Square - Lab Architecture Studio



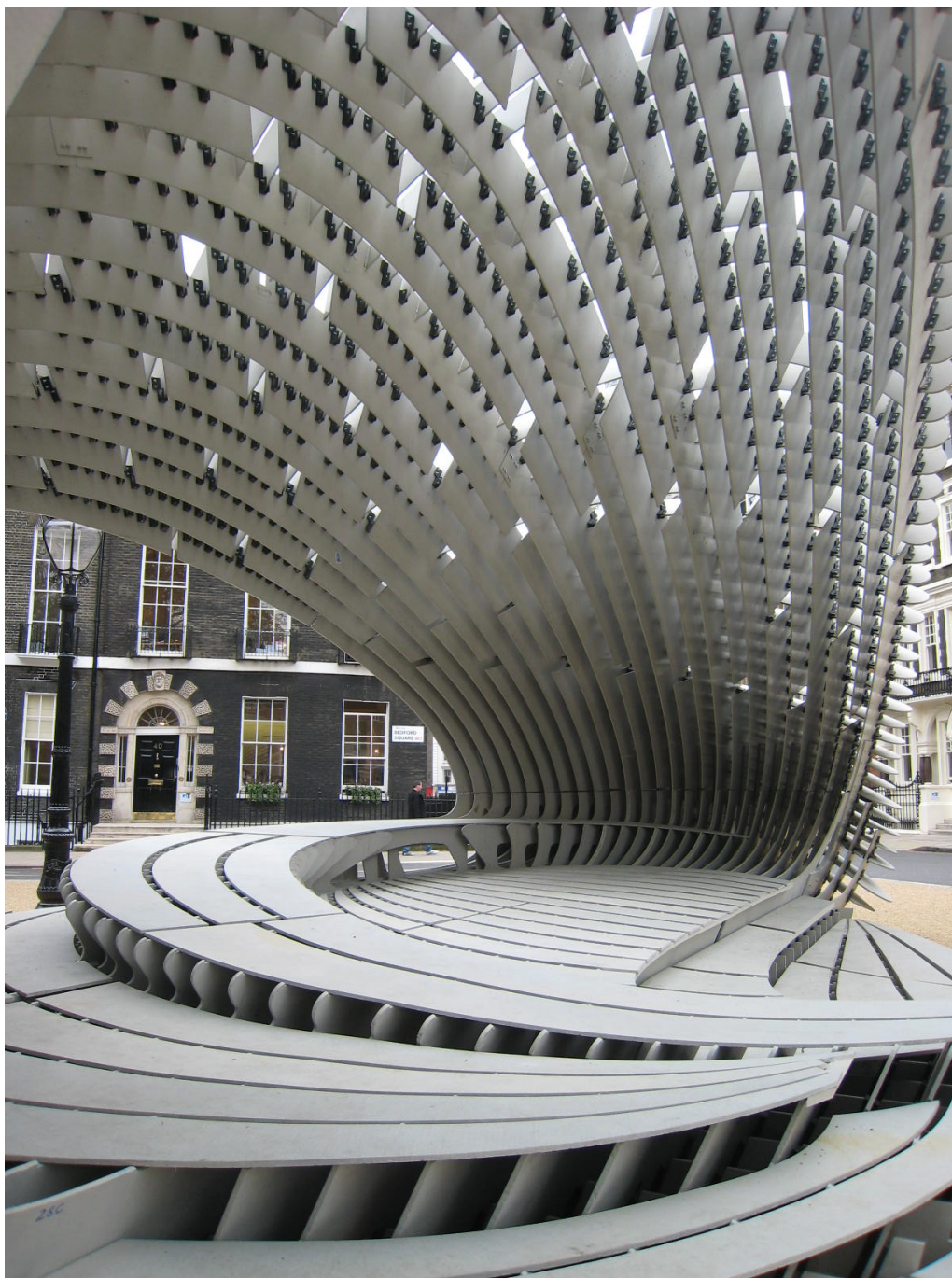
Fonte:upload.wikimedia.org - <[https://upload.wikimedia.org/wikipedia/commons/5/5d/Federation_Square_\(SBS_Building\).jpg](https://upload.wikimedia.org/wikipedia/commons/5/5d/Federation_Square_(SBS_Building).jpg)>

Figura 3 – Pavilhão realizado em uma pesquisa na Universidade de Stuttgart-ICD-ITKE 2014



Fonte:www.ehu.eus - <<http://www.ehu.eus/ehusfera/industrialized-architecture/files/2015/09/ICD-ITKE-Research-Pavilion-2014-15-1-728x406.jpg>>

Figura 4 – Pavilhão realizado em uma pesquisa na Architectural Association [C]Pavilion AA-DRL 2007



Fonte: upload.wikimedia.org - <<https://upload.wikimedia.org/wikipedia/commons/1/16/Aapavilion.JPG>>

Figura 5 – Consultoria de Design to production - Mercedes Benz Museum



Fonte: upload.wikimedia.org - <https://upload.wikimedia.org/wikipedia/commons/1/1a/Mercedes-Benz_Museum_201312_02_sunset.jpg>

Figura 6 – Consultoria de Programming architecture - Shenzhen airport



Fonte: c2.staticflickr.com <Fonte:https://c2.staticflickr.com/6/5533/11447235775_fb6f3abc08_b.jpg>

O avanço da computação pós-Segunda Guerra Mundial atingia todos os campos do conhecimento. Dentre os novos métodos computacionais que começaram a ser desenvolvidos na década de 50, alguns buscavam compreender e simular a capacidade generativa da natureza, enquanto outros tentavam compreender e mimetizar características cognitivas do ser humano (RUSSELL *et al.*, 1995).

Esse desenvolvimento culminou em um novo campo de pesquisa, denominado Inteligência Artificial, termo criado por John McCarthy em 1956 em um workshop em Boston. Atualmente a definição de Inteligência Artificial varia entre duas dimensões (RUSSELL *et al.*, 1995). A primeira é centrada no ser humano, uma ciência empírica, com hipóteses e experimentos. A segunda é voltada para a racionalidade, e combina matemática e engenharia (RUSSELL *et al.*, 1995), sendo essa última a que mais se aproxima dos processos e métodos desenvolvidos ao longo dos anos em projeto computacional.

Entretanto, o projeto computacional não é um campo novo de pesquisa, pois essa temática vem sendo discutida desde o início dos anos 60, quando os primeiros software de *computer aided-design* (CAD) estavam no início de seu desenvolvimento (ROCHA, 2004). Uma das metodologias do design computacional que foi amplamente discutida paralelamente ao desenvolvimento das ferramentas CAD foi o *Space Planning* (GRASON, 1971; MILLER, 1970; EASTMAN, 1971; EASTMAN, 1973).

Space Planning (SP) é um campo de pesquisa que estuda o processo de arranjo de uma série de elementos topológicos e/ou geométricos, em um determinado espaço, em que distância, adjacência e outras funções de arranjo são o principal objetivo (EASTMAN, 1971; EASTMAN, 1973). Geralmente os problemas em SP são definidos através de dois conceitos: (1) problemas de topologia e (2) restrições geométricas. Primeiramente as restrições topológicas definem uma hierarquia de relações espaciais, como adjacências, não adjacências, proximidades e profundidade entre os elementos espaciais (HOMAYOUNI, 2007). As restrições geométricas são definidas através de superfícies, dimensões, orientação espacial, entre outros métodos.

Os anos entre 1970 e 1980 foram particularmente produtivos para a produção de pesquisa em SP (GRASON, 1971; MILLER, 1970; LIGGETT; MITCHELL, 1981). Nesse período, por exemplo, algumas pesquisas em SP foram relacionadas com a teoria dos grafos, teoria matemática que descreve relações topológicas através de vértices e arestas. Toda essa discussão aconteceu anteriormente à popularização das ferramentas CAD entre os arquitetos e designers.

Em 1975, Mitchell escreveu um trabalho seminal para a fundação do projeto computacional, “*The theoretical foundation of computer-aided architectural design*” (MITCHELL, 1975). Nesse trabalho, o autor conceituou o processo de projeto como um tipo específico de processo de solução de problema (ROCHA, 2004). Em 1964, a conferência “*Architecture and the Computer*”, realizada em Boston, havia contado com a participação

de arquitetos, cientistas da computação, urbanistas e economistas, discutindo as novas possibilidades do uso do computador em seus respectivos campos de pesquisa (ROCHA, 2004). Dessa conferência participaram grandes nomes como Marvin Minsky, cientista cognitivo cofundador do instituto de inteligência artificial do MIT, pioneiro nesse campo de pesquisa. Também participaram arquitetos consagrados, como Christopher Alexander, pioneiro nos estudos de *Design Methods* e Walter Gropius, que 30 anos antes havia sido essencial para a construção do pensamento moderno em arquitetura (ROCHA, 2004).

No início dos anos 90, muitas pesquisas em projeto computacional foram publicadas relacionando métodos de projeto com inteligência artificial. Essas pesquisas buscavam discutir mecanismos generativos, que funcionavam como intermediador entre arquiteto e o sistema generativo, num processo colaborativo de tomada de decisões de projeto, ampliando assim o potencial de cálculo das variáveis arquitetônicas e sugerindo novas possibilidades, inimagináveis sem o uso do computador, estendendo a capacidade humana de processamento de informação (TERZIDIS, 2006).

Posteriormente, a utilização desses mecanismos foi definida por Rivka Oxman como *Generative models* (OXMAN, 2006). Para melhor compreender onde os *Generative Models* se enquadram no design computacional, será apresentado o quadro teórico que Oxman delineou em “*Theory and design in the first digital age*” (OXMAN, 2006). São apresentados a seguir os cinco modelos do que chamou de *Digital Design*, termo criado no início da década de 90, resultado de uma série de experiências no campo da metodologia de projeto e processos construtivos. Nesse campo, o computador é parte essencial no processo de conceber o projeto e de se produzir a obra de arquitetura através de métodos de fabricação digital.

Sem ater-se à análise dos graus de interação das características metodológicas do ato de projetar definidos por Oxman, a abordagem desse trabalho será de como os *Evolutionary design models* podem ser explorados para a resolução de problemas de *space planning*, buscando um melhor desempenho funcional e qualitativo do espaço.

Os modelos delineados por Oxman são apresentados no quadro 1.1

Quadro 1.1: Modelos de *Digital Design* delineados por Oxman

<i>CAD Models</i> Modelo que marca o início da transferência da plataforma de desenho tradicional representativo (o desenho de prancheta), para o computador.	<i>CAD descriptive model</i> Caraterizado pela manipulação de representações gráficas de objetos digitais.	<i>Generation-evaluation predictive CAD</i> Modelos que usualmente estão associados com estimativas de custo, comportamento estrutural e desempenho ambiental.
<i>Digital Formations Models</i> Modelo onde sistema CAD não é mais visto como uma prancheta eletrônica, e sim como participante ativo no processo de tomada de decisões do projeto, sendo um modelo digital interativo de formação do objeto.	<i>Topological Formation Models</i> São explorados conceitos relacionados à topologia (estudos das estruturas relacionais dos objetos em vez de sua geometria mensurável). Esse modelo <i>CAD</i> é desenvolvido através de métodos de modelagem baseada em geometrias <i>NURBS</i> (<i>non-uniform rational bsplines</i>), que torna possível a concepção de formas geométricas complexas de maneira interativa.	<i>Associative Design</i> Baseado na concepção de modelos <i>CAD</i> paramétricos, ou seja, que exploram técnicas de geometria associativa. Nesse modelo, as relações entre os objetos são descritas de maneira explícita, estabelecendo a interdependência de vários objetos
<i>Motion-based Formation Models</i> Explora as propriedades dinâmicas do espaço e dos objetos para a geração da forma, utilizando técnicas de animação.		
<i>Generative Models</i> São modelos caraterizados por mecanismos computacionais nos processos de geração da forma. Nesse modelo o designer interage com o mecanismo generativo, que passa a ser um intermediador no processo de tomada de decisões do projeto.	<i>Shape Grammars</i> Mecanismo generativo baseado em uma lógica de regras de composição formal.	<i>Evolutionary design models</i> A forma emergente e decorrente do resultado de um processo evolutivo. Nesse processo são utilizados métodos como o algoritmo genético, que é empregado em um problema específico a fim de buscar uma solução otimizada .
<i>Performance Models</i> Modelos que utilizam tecnologias digitais para geração da forma, buscando resultados que atendam características relativas ao desempenho do objeto. Essas características poderão ser condicionadas a distintas problemáticas, como: condições ambientais, programa, território, entre outros.	<i>Performance-based Formation Models</i> Busca atender a resultados de performance baseando-se em parâmetros de distintas problemáticas (como as descritas acima), de maneira associativa, empregando técnicas de simulação que produzem expressões paramétricas de desempenho.	<i>Performance-based Generation Models</i> Esse modelo é baseado em processos generativos de geração da forma, destinados à obtenção de um resultado que atenda às questões relativas ao desempenho do objeto.
<i>Integrated Models</i> Compreende a integração dos distintos processos de geração da forma: Formação, Geração, Avaliação e Desempenho.		

1.2 Problemática e Justificativa

A problemática e justificativa desta pesquisa estão relacionadas à potencialidade dos estudos dos algoritmos evolutivos para otimização de problemas de *layout*. Atualmente vivemos um contexto pós-industrial, com a mudança de um paradigma de produção em massa padronizada para um de produção em massa personalizada (*mass customization*). A abordagem evolutiva por um lado permite gerar variações combinatórias com características semelhantes, e por outro selecionar aquelas que melhor atendem aos requisitos especificados.

Nos anos 50, o psicólogo Joy Paul Guilford fez a distinção entre pensamento divergente e convergente (GUILFORD, 1967), sendo o pensamento divergente a habilidade de encontrar o número máximo de respostas possíveis a um problema particular, e o pensamento convergente a habilidade de encontrar a melhor resposta para determinado problema. A abordagem evolutiva voltada à produção em design é desenvolvida justamente dentro desse quadro teórico, em um processo cíclico entre divergência e convergência, pois ao mesmo tempo que gera uma família de soluções possíveis a cada geração, busca a melhor resposta dentre a população através da função objetivo (*fitness function*).

A aplicação de sistemas evolutivos em arquitetura pode ser uma dentre outras maneiras de atender a esse novo paradigma de desejo de exclusividade e produção da sociedade atual, pois através dos AE (Algoritmo Evolutivo) torna-se possível produzir, através de um processo generativo, um conjunto de objetos/componentes arquitetônicos, que respondam a necessidades e desejos singulares, além da nova agenda ambiental onde desempenho em arquitetura torna-se um item indispensável diante do paradigma de sustentabilidade, que interfere na sociedade no âmbito econômico, social e ecológico.

Os escritórios de arquitetura e urbanismo, no seu dia-a-dia, frequentemente se deparam com problemas relacionados a como arranjar funcionalmente e qualitativamente os espaços, seja na composição de *layouts* internos ou no arranjo de edifícios no espaço urbano.

Nos últimos anos, novas ferramentas gráficas vêm popularizando as metodologias do projeto computacional, e assim, os escritórios estão buscando cada vez mais aplicar os métodos relacionados a esse campo de pesquisa.

Esta pesquisa busca responder, através dos procedimentos metodológicos adotados, duas questões :

- (i) As novas ferramentas computacionais gráficas poderiam auxiliar na popularização e divulgação dos métodos de SP ?
- (ii) Os métodos de SP poderiam ser utilizados no dia-a-dia dos escritórios de arquitetura , como processo auxiliar no processo de arranjo e composição espacial ?

A **hipótese** desta pesquisa consiste na afirmação de que as novas ferramentas de apoio ao projeto de arquitetura (CAAD) , assim como estão auxiliando na divulgação e aproximação de arquitetos de uma série de metodologias do projeto computacional, que anteriormente estavam restritas aos meios acadêmicos e também poderiam auxiliar na divulgação dos métodos de SP, tanto no meio acadêmico quanto no meio profissional, e os métodos de automatização de SP poderiam ser explorados no processo de arranjo e composição dos espaços nos escritórios de arquitetura, sendo que todo esse processo poderia ser potencializado pela criação de ferramentas ou algoritmos específicos de *Space Planning*.

Este trabalho teve uma abordagem teórico-prática do assunto, ou seja, buscou desenvolver um conteúdo teórico sobre *Space Planning* e AE, desenvolvendo experimentos e exercícios de projeto que abordaram estas metodologias.

A principal contribuição deste trabalho foi corroborar a hipótese acima, verificando que as novas ferramentas computacionais de auxílio ao projeto podem contribuir de maneira efetiva em determinadas situações de projeto, sendo que, os novos ambientes de interface "amigável" podem ser facilitadores da aproximação de arquitetos e antigas metodologias do design computacional que tenderam a ficar esquecidas devido a sua complexidade.

1.3 Objetivos

Geral

Entender e explorar as possibilidades da resolução de problemas de *space planning* (geração de *layouts* automatizada) através dos princípios da computação evolutiva.

Específicos

1. Desenvolver um conteúdo teórico crítico sobre *space planning* em arquitetura e urbanismo sob a ótica dos AE.
2. Identificar processos, experimentos e projetos que utilizaram como metodologia algoritmos evolutivos na resolução de problemas de *space planning*.
3. Definir um conjunto de problemas de projeto voltados à otimização topológica e geométrica do programa de necessidades e de sua implantação no território, que permitam desenvolver experimentos com o uso de AE, demonstrando sua utilidade.
4. Desenvolver algoritmos que simulem esses processos evolutivos na resolução de problemas de *space planning*, voltados à otimização funcional e qualitativa do espaço, entendendo a avaliação qualitativa do espaço como temáticas compositivas e perceptivas relacionadas à leitura do espaço.

5. Organizar e realizar um workshop para verificar a viabilidade de aplicação do método por jovens arquitetos no Brasil.

1.4 Relação com outras pesquisas

Recentemente foi desenvolvida uma pesquisa no programa de pós graduação ATC (Arquitetura, Tecnologia e Cidade), sob orientação de Gabriela Celani, pelo doutor Jarryer Andrade Martino, intitulada “Algoritmos Evolutivos como método para desenvolvimento de projetos de arquitetura e urbanismo”, que teve por objetivo :

“Estudar os algoritmos evolutivos, organizar seu conteúdo teórico em um banco de dados para consulta, identificar uma ferramenta computacional que permitirá a implementação do método, identificar as situações concretas em que os arquitetos e urbanistas possam utilizá-las, e implementar o método evolutivo através de workshop a ser realizado com alunos do curso de Arquitetura e Urbanismo da UNICAMP como forma de verificação e validação.”(MARTINO, 2015).

A pesquisa de Martino apresentou um panorama dos algoritmos evolutivos e criou um banco de dados de situações concretas de utilização. O trabalho aqui realizado pretende ser complementar ao de Martino, focando em uma das vertentes pertencentes aos algoritmos evolutivos, descrita por Bentley como *Integral Evolutionary Design*, que corresponde à intersecção de *Evolutionary Design Optimisation* e *Creative Evolutionary Design* (BENTLEY, 1999), ou seja, esta pesquisa buscou se basear na gênese criativa da forma, com ênfase em desempenho, através de exercícios de projeto direcionados aos problemas de *space planning*, documentando os produtos do processo de projeto.

1.5 Metodologia

Dadas as características das perguntas que esta pesquisa buscou responder, e os objetivos que foram delineados, sentiu-se a necessidade de construir um conhecimento base, tanto sobre as metodologias empregadas nos problemas de *space planning*, quanto sobre os princípios básicos de funcionamento dos algoritmos evolutivos, afim de poder explicitar os procedimentos envolvidos nessas temáticas.

Esta pesquisa adotou como base metodológica a **pesquisa exploratória**, e teve uma abordagem teórico-prática do assunto, ou seja, buscou desenvolver um conteúdo teórico sobre as metodologias do *space planning* com o desenvolvimento de um exercício prático, como também um conteúdo teórico sobre algoritmos evolutivos.

De acordo com (GIL, 2002) a pesquisa exploratória tem como objetivo construir uma familiaridade com o objeto de pesquisa a fim de torná-lo mais “explícito”, e seus métodos são bastante flexíveis e variam de acordo com o caso estudado.

Os procedimentos metodológicos adotados foram nesta pesquisa : **levantamento bibliográfico, métodos experimentais, coleta de dados e prova de conceito.**

O **levantamento bibliográfico** se deu em três etapas, que foram divididas por temáticas:

- levantamento bibliográfico sobre algoritmos evolutivos
- levantamento bibliográfico sobre as metodologias do *space planning*
- levantamento bibliográfico sobre publicações de *space planning* com abordagem evolutiva

O foco da revisão da literatura foi especificamente a resolução de problemas de *space planning* através de algoritmos evolutivos, uma vez que um estudo mais amplo sobre algoritmos evolutivos foi desenvolvido por outro membro da equipe, Jarryer Martino. As referências de *space planning* com a abordagem evolutiva encontram-se principalmente em anais de congressos, por se tratar de um assunto ainda muito especulativo e com poucas aplicações construídas. Foram realizados levantamentos em bases de dados online (*CUMINCAD*, *IEEXplore*, *ACM digital Library*, *Research Gate*, *Science direct* e *CiteSeerX*) e foram selecionados 31 artigos, publicados entre 1992-2014.

Os **métodos experimentais** foram aplicados através da implementação de algoritmos experimentais sobre *space planning*, foi escolhido um dos métodos estudados na revisão de literatura, a escolha se deu com base nas possibilidades abrangentes que o método abarcava. Foram explorados experimentos em nível incremental de complexidade, aplicados em duas problemáticas arquitetônicas: o planejamento de um *layout* de uma residência e o planejamento espacial de edifícios em uma quadra urbana, sendo que o nível incremental de complexidade variou de acordo com o número de variáveis aplicadas a cada exercício. Estas temáticas foram escolhidas por serem tópicos frequentes no dia-a-dia dos escritórios de arquitetura.

Esta pesquisa é complementar ao trabalho de Martino desenvolvendo e implementando algoritmos evolutivos também em Galápagos, mas com o objetivo específico de resolução de problemas de *space planning*. Como método de pesquisa, este trabalho implementou códigos com nível de complexidade incremental. Os critérios de escolha no desenvolvimento do algoritmo foram determinados pela eficiência da ferramenta em atender de maneira clara e coesa a natureza do problema. (*Rhinoceros 3d* - *Grasshopper* - *Galápagos*) e não apenas na velocidade de solução (desempenho), mas também buscou contribuir para a inteligibilidade do processo. Os algoritmos foram desenvolvidos em linguagem de programação visual juntamente com *scripts*. A linguagem de programação escolhida para a implementação dos *scripts* foi *Python*. A pesquisa buscou problemáticas arquitetônicas

passíveis de implementação com algoritmos evolutivos na resolução de problemas de *space planning*, por exemplo, que incluíam em seu *briefing* objetivos conflitantes de difícil otimização. Alguns exemplos típicos são problemas de distribuição de serviços no desenho urbano de uma quadra ou conjunto de quadras, que devem estar adjacentes ou não uns aos outros, formando *clusters* ou afastando-se, com fluxos de distribuição entre eles de serviços, materiais e pessoas. Outra possibilidade explorou o desenvolvimento de *layouts* arquitetônicos. A partir dos casos identificados nessas problemáticas foram desenvolvidos **experimentos** de *Space Planning* que foram otimizados através do Galápagos.

A **prova de conceito** foi realizada através de um Workshop realizado na UNICAMP, que teve por objetivo coletar dados qualitativos a respeito do método buscando responder as questões da pesquisa. O workshop foi montado com base nos algoritmos experimentais que foram desenvolvidos na etapa dos métodos experimentais. Foram selecionados participantes que já tinham um nível intermediário no uso da ferramenta base de programação gráfica escolhida (*Grasshopper*) para o desenvolvimento dos algoritmos, com o intuito que pudessem colaborar na identificação de problemas, propostas de melhoramento, sugestões, etc. Com a realização do workshop foi possível realizar uma prova de conceito qualitativa e levantar dados qualitativos sobre o método adotado, dentro das restrições que delimitavam o quadro amostral dos participantes.

A produção obtida através dos resultados verificou a eficiência dos métodos desenvolvidos e estabeleceu um *feedback* para possíveis correções e adaptações, possibilitando, dessa maneira, a verificação de assimilação dos conceitos abordados após o desenvolvimento de uma sistemática de aplicação dos algoritmos na metodologia. Os *workshops* têm sido um recurso muito utilizado para exploração de novos conceitos em arquitetura. Um exemplo é o formato utilizado pelo congresso *Smart Geometry*, em que em vez da seleção e apresentação de artigos, os candidatos à participação no congresso propõem o oferecimento de *workshops* que discutem, de maneira prática, a aplicação de pesquisas em andamento.

1.6 Estrutura da dissertação

Esta dissertação está dividida em 8 capítulos. O primeiro capítulo é a **introdução**, em que se apresenta a problemática da pesquisa e se definem os objetivos. O segundo apresenta os métodos empregados nos problemas de *space planning*. O terceiro capítulo apresenta alguns dos **métodos evolutivos** utilizados na computação e discute experimentos realizados por arquitetos. O quarto capítulo apresenta uma revisão da literatura de artigos relacionados a problemas de *space planning* com **abordagem evolutiva**. O quinto capítulo apresenta os materiais e métodos que foram utilizados nos experimentos. O sexto capítulo apresenta os **experimentos** que foram realizados. O sétimo capítulo

apresenta e descreve o **workshop** realizado e os procedimentos utilizados no mesmo. O oitavo e último capítulo é o fechamento desta dissertação e são as **considerações finais**.

Parte I

Space Planning

2 *Space Planning*

Uma das tarefas importantes que arquitetos e designers resolvem com frequência em projetos de arquitetura é a organização do espaço, que deve ser criativa e funcionar de maneira apropriada atendendo a um determinado propósito. O processo de uma lista de elementos espaciais é o principal objetivo de um ramo de pesquisa conhecido como *Space planning* (SP), sendo que os problemas a serem solucionados estão relacionados com distância, adjacência e outras funções de arranjo (EASTMAN, 1973).

Os problemas de SP não são exclusividade do projeto arquitetônico, pois fazem parte de um amplo campo de pesquisa que atinge uma grande variedade de problemas, como placas de circuitos impressos, teclados, design de turbinas hidráulicas, posicionamento de máquinas industriais, etc (SINGH; SHARMA, 2006). Este trabalho, por se tratar de um estudo no campo da arquitetura, tratará dos problemas de SP relacionados ao espaço arquitetônico.

Liggett (2000) define a divisão dos métodos de *space planning* em duas categorias: **métodos de representação do espaço** e **métodos de avaliação do espaço**. Dentre os métodos de representação do espaço identificados por Liggett (2000) temos : **atribuição um a um** (*one-to-one assignment*), **empilhamento** (*stacking*) e **agrupamento** (*blocking*).

Atribuição um a um (*“One-to-one Assignment”*)

O método de **atribuição um a um** (*one-to-one assignment*) consiste na alocação de uma lista de ambientes em uma lista de locais (LIGGETT, 2000). Existem dois principais métodos para essa abordagem: o método de **áreas iguais** e o de **áreas desiguais**.

No sistema de **áreas iguais**, um plano é subdividido em partes iguais, de forma que cada parte represente um ambiente, enquanto no método de **áreas desiguais** um plano é subdividido em pequenas unidades, sendo que o ambiente é criado a partir da soma das unidades, de acordo com as áreas requeridas.

Empilhamento (*“Stacking”*)

O método de **empilhamento** (*“Stacking”*) é semelhante ao método de atribuição um a um, no modo em que subdivide o espaço, mas diferentemente do método anterior, as funcionalidades dos espaços podem ser sobrepostas em um espaço ou uma funcionalidade, podendo fazer parte de um ou mais espaços, ou seja, cada unidade da subdivisão pode receber uma ou mais funcionalidades, como também, cada funcionalidade pode ser atribuída a uma ou mais unidades.

Agrupamento (“Blocking”)

No método de **agrupamento** (*blocking*) um polígono representa um ambiente que deve ser capaz de manter suas características geométricas e ao mesmo tempo se ajustar com outros ambientes em um determinado perímetro, de acordo com regras pré-determinadas.

Os métodos de avaliação do espaço podem ser divididos em 3 grandes classes de acordo com Liggett (2000): **Otimização dos custos** de um único critério, **otimização através de grafos** e **otimização de restrições personalizadas**.

Otimização do custo de materiais

A otimização de uma única função objetivo está especificamente direcionada para a minimização de custos, que podem ser relacionados à comunicação entre atividades ou ao fluxo de materiais entre ambientes de um *layout*.

Otimização através de grafos

O método de otimização através de grafos é baseado na teoria matemática dos grafos. A premissa principal deste método é gerar *layouts* que respeitem restrições relacionadas a critérios de adjacência previamente definidos.

O grafo estabelece relações topológicas entre ambientes de um programa de necessidades, variando em escala, podendo por exemplo, se referir à distribuição de edifícios por quadras em um desenho urbano ou à distribuição de ambientes por um *layout* arquitetônico. Através dessas distribuições são definidas relações de adjacência e não adjacência entre os espaços, sendo que essas relações podem ter um espaço geométrico definido ou apenas se restringir às relações topológicas, permitindo assim a criação de espaços com formas geométricas distintas, mas que mantêm as propriedades de uma mesma relação topológica, nas plantas residenciais do arquiteto Frank Lloyd Wright (figura 7). A subseção 2.2.2 descreverá mais detalhadamente o conceito de grafos.

Otimização de restrições personalizadas

No método de otimização de restrições personalizadas são desenvolvidas uma série de restrições e critérios de relação entre atividades de um *layout*, e que são estabelecidos de acordo com a aplicabilidade desejada pelo projetista.

Dada a quantidade de variáveis a que está sujeito o projeto arquitetônico e urbanístico, automatizar a geração de soluções, de modo que atenda a todos os parâmetros necessários em um projeto torna-se um objetivo de difícil resolução para o arquiteto. Além disso, a seleção de variáveis adotada pelo projetista, dentre uma geração de possíveis soluções, geralmente utiliza um critério ambíguo, nebuloso, onde por muitas vezes encontram-se implícitas as preferências pessoais do projetista, que não estarão explícitas no sistema generativo (MITCHELL, 1975).

Figura 7 – Diferentes configurações de plantas residências projetadas por Wright com uma mesma relação de topológica entre os ambientes



Fonte: "A Lógica da Arquitetura"(MITCHELL, 1990)

Os problemas de *space planning*, quando entendidos de maneira holística, como relatado por Mitchell, foram amplamente discutidos por outros pesquisadores e definidos como problemas bem definidos (*ill-defined*) ou mal definidos (*ill-structured*) (REITMAN, 1965; EASTMAN, 1969; UHR, 1973).

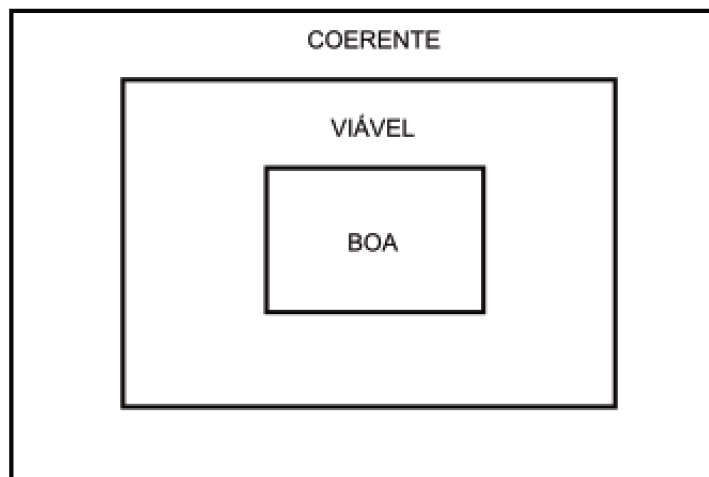
Os problemas bem definidos são aqueles absolutamente corretos e com soluções conhecidas. São conhecidos por duas características : (1) existe apenas uma solução correta para o problema e esta pode ser determinada com total certeza (2) é garantido um procedimento para atingir esta solução.

Problemas mal definidos são aqueles em que as respostas são conflituosas, tendendo a diferentes soluções. Estes problemas podem ter múltiplas soluções, como também nenhuma solução, e para esse tipo de problema não é garantido um procedimento que atinja estas soluções.

Hebert Simon Simon (1975) argumenta que não existem diferenças substanciais entre problemas definidos como "bem definidos" e "mal definidos", e argumenta que programas que são modelados para solucionar problemas bem definidos não diferem em princípios de programas que resolvem problemas mal definidos. Algoritmos de processos generativos de projeto sempre estarão sujeitos a escolhas implícitas na modelagem e implementação de um sistema, assim como os projetistas, independentemente das ferramentas que utilizem, do lápis ao computador, hierarquizam as variáveis arquitetônicas e optam por solucionar determinadas questões em detrimento de outras que julgam menos relevantes. Esse processo não inviabiliza o desenvolvimento do projeto, sendo que a qualidade arquitetônica, nesse caso, passa a ser definida pelo conjunto de variáveis escolhidas pelo projetista. A solução final do projeto deixa de ser uma "solução" ótima em todos

os critérios e passa a ser compreendida como uma opção viável (figura 8) (MITCHELL, 1990).

Figura 8 – Soluções de projeto de acordo com o objetivo



Fonte: Adaptado de "A Lógica da Arquitetura"(MITCHELL, 1990)

2.1 Os Algoritmos de Space Planning

Modelos que permitem a análise do *layout* como mecanismo para gerar soluções são estudados desde o final dos anos 50 (HERAGU, 2008). Essas primeiras pesquisas envolviam o uso de gráficos de fluxos, gráficos de processo e a experiência e conhecimento do analista de *layout*. Foi com o nascimento da **pesquisa operacional**, no início dos anos 50, que foi iniciado o desenvolvimento de modelos analíticos para projeto e planejamento de espaços (HERAGU, 2008).

O primeiro algoritmo para os métodos de análise desenvolvido foi o *quadratic assignment problem* (QAP) (KOOPMANS; BECKMANN, 1957), para solucionar problemas que abrangiam largas escalas, como a distribuição de depósitos por uma grande região geográfica, para projetos de menor porte como um edifício ou a distribuição de *layouts* por uma fábrica. O modelo QAP foi desenvolvido visando diminuir o custo de transporte do fluxo de materiais que acontecem entre uma série de elementos (ambientes) que estão inter-relacionados. O problema matemático pode ser formulado da seguinte maneira (KOOPMANS; BECKMANN, 1957):

$$(\sum a_{ij} + \sum \sum q_{i1i2} c_{j1j2})$$

Onde:

a_{ij}	custo fixo de locação de um ambiente i para um ambiente j
c_{j1j2}	distancia de um local $j1$ para um local $j2$
i	unidade de ambiente a ser alocada
j	possível localização

q_{i1i2} interações entre unidades de ambiente $i1$ e unidades de ambiente $i2$

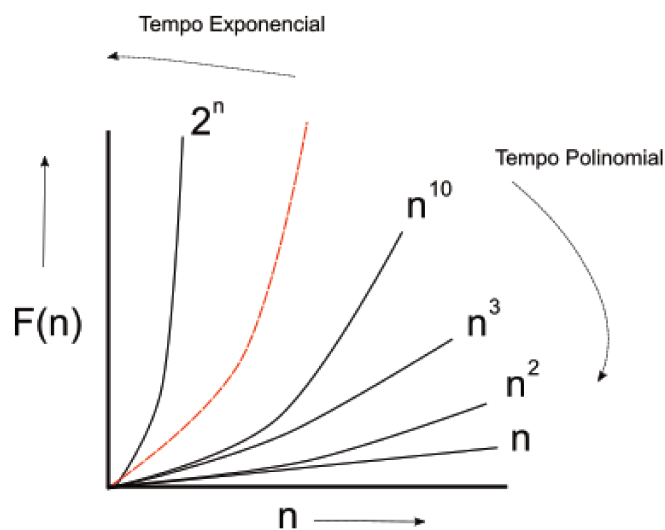
Desde o modelo de QAP, muitos algoritmos foram desenvolvidos para a resolução e otimização de problemas de SP. O processo de otimização de um problema através de algoritmos busca responder da melhor forma possível a um dado problema. Para resolvê-lo é formulada uma **função objetivo**, que associa para cada solução de um espaço solução um número real, permitindo desta maneira medir a qualidade da resposta ao problema. A resposta dada pela função objetivo pode ser formulada para a maximização do valor do número real (quanto maior o valor do número melhor a solução) ou minimização do valor (quanto menor o valor melhor a solução).

Os métodos de otimização podem ser definidos como determinísticos ou estocásticos. Os métodos determinísticos são aqueles que sempre darão a mesma resposta partindo de um mesmo ponto inicial, enquanto os métodos estocásticos utilizam números randômicos no processo que modificam o processo de escolha das alternativas, acarretando em diferentes respostas a cada execução do algoritmo.

2.2 Algoritmos Heurísticos

Os algoritmos heurísticos são modelados para encontrar uma solução viável e aceitável, de acordo com um tempo computacional coerente. Os algoritmos heurísticos dedicados a resolver problemas de SP podem ser classificados dentro de três categorias (HERAGU, 2008):

Figura 9 – Relação entre tempo computacional e dados de entrada



Fonte: Adaptado de ftc Unesp <<http://docs.fct.unesp.br/docentes/dmec/olivete/tc/arquivos/Aula11.pdf>>

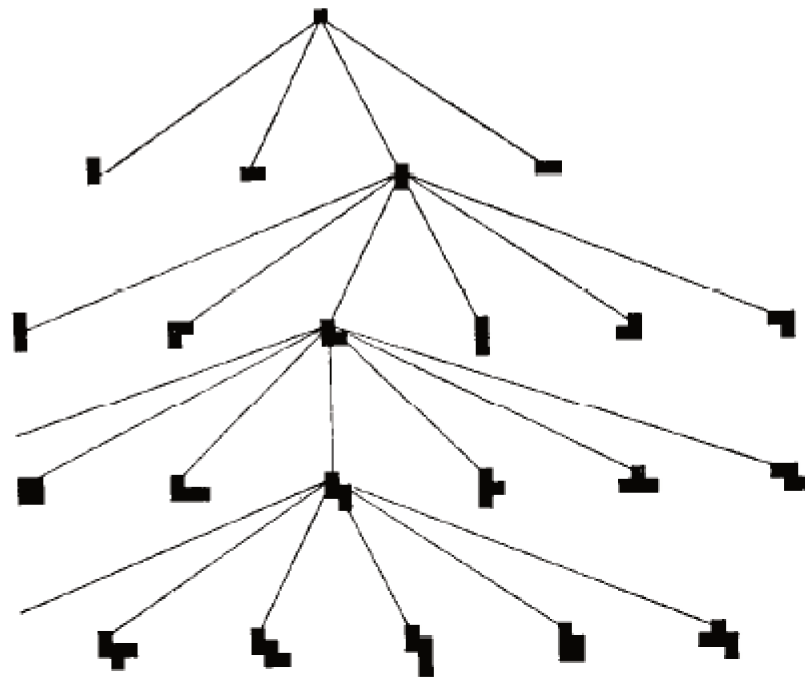
- *Construction Algorithms*
- *Graph theoretic approach*
- *Improvement Algorithms*

2.2.1 Construction Algorithms

Os *construction algorithms* são algoritmos heurísticos que iniciam seus procedimentos com um espaço de soluções vazio para um dado *layout* e vão adicionando cada ambiente (ou uma lista de ambientes) um após o outro até que o *layout* esteja completo. As principais diferenças entre os vários *construction algorithms* estão relacionadas com o critério usado para determinar o primeiro elemento a ser colocado no layout, a subsequência de elementos adicionados ao *layout* e o local do primeiro ou da subsequência de elementos do layout (HERAGU, 2008):

Os *construction algorithms* geralmente são concebidos através de uma estrutura em árvore, em que a raiz do problema é a situação inicial do projeto, enquanto os nós intermediários são representações parciais da solução e os nós terminais são as representações finais da solução (figura 10).

Figura 10 – Distribuição em árvore dos construction algorithms



Fonte: “Techniques of automated design in architecture: A survey and Evaluation” (MITCHELL, 1975)

Tabela 1 – Representação e valores da taxa de proximidade

Letras	Valores exponenciais	Valores
A	4^3	64
E	4^2	16
I	4^1	4
O	4^0	1
U	0	0
X	-4^5	-1,024

Fonte: Adaptado de "*Facility Planning and Layout Design*" (HIREGOUDAR, 2007)

Esses algoritmos em problemas de SP geralmente adotam uma orientação global ou local diante de um problema. Os algoritmos locais consideram somente as alocações que já foram feitas, e seu custo computacional é baixo, porém as soluções encontradas tendem a ser de baixa qualidade, enquanto os algoritmos globais avaliam os possíveis movimentos futuros de uma possível alocação, mas esses algoritmos são mais custosos computacionalmente, entretanto, produzem soluções melhores que os algoritmos locais (LIGGETT, 2000).

Um exemplo de implementação de *construction algorithm* é o Software ALDEP (*Automated Layout Design Program*), foi desenvolvido na IBM em 1967 por Seehof e Evans. O ALDEP é geralmente usado quando as relações de adjacência entre os ambientes é o principal objetivo. O software vai alocando os ambientes no layout um após o outro, através de uma taxa de proximidade que varia de A a X (tabela 1) e também áreas requeridas. O processo continua até que todos os ambientes sejam inseridos no *layout*. A pontuação do *layout* é feita através da soma dos valores numéricos atribuídos a taxa de proximidade. Os valores usados pelo ALDEP são apresentados na tabela 1.

O ALDEP pode alocar até 63 ambientes e pode gerar layouts de até 3 andares (HIREGOUDAR, 2007). O ALDEP funciona de acordo com os seguintes passos:

Passo 1:

São inseridos os seguintes dados de entrada:

1. Comprimento e altura do ambiente
2. A área de cada ambiente
3. O valor de *Minimum Closeness Preference* (MPC)(distância mínima de ambiente para ambiente)
4. O comprimento da varredura (quantidade das iterações)
5. A tabela de relações com a taxa de proximidade
6. Localização e tamanho de qualquer área restrita de movimentos

Passo 2:

O programa seleciona randomicamente um ambiente e o aloca no *layout*.

Passo 3:

O algoritmo utiliza o *minimum closeness* para determinar qual ambiente será alocado de acordo com o ambiente anterior. O algoritmo seleciona o ambiente que tem a maior taxa de proximidade, caso não exista um ambiente que tenha o *minimum closeness* requerido, qualquer um dos ambientes restantes é alocado.

Passo 4:

Se todos os ambientes foram alocado vá para passo 5, caso contrário volte para passo 3.

Passo 5:

O algoritmo computa a pontuação obtida pelo *layout*.

Passo 6:

Se o valor da pontuação obtida é aceitável vá para passo 7, caso contrário volte para passo 2 .

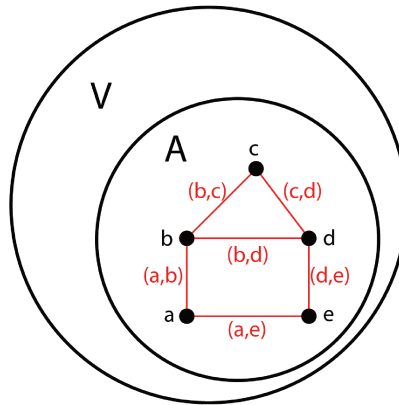
Passo 7:

Imprima o *layout* e a pontuação correspondente.

2.2.2 Graph theoretic approach

Para entender o método de *graph theoretic* devemos primeiramente entender os conceitos de grafo plano (figura 12) e grafo planar máximo (figura 13). Um grafo é um par (V, A) de vértices e arestas, sendo V um conjunto não ordenado de vértices e A um subconjunto de V , em que os vértices são conectados através de arestas (figura 11) (FEOFILOFF *et al.*, 2011).

Figura 11 – Grafo (V,A)

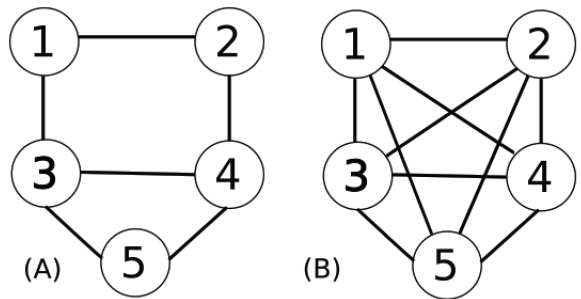


Fonte: Figura Produzida pelo autor

Um grafo é planar se é desenhado em duas dimensões sem que haja qualquer tipo de intersecção; é planar máximo, se todas as interligações possíveis estiverem feitas sem que haja qualquer intersecção de arestas.

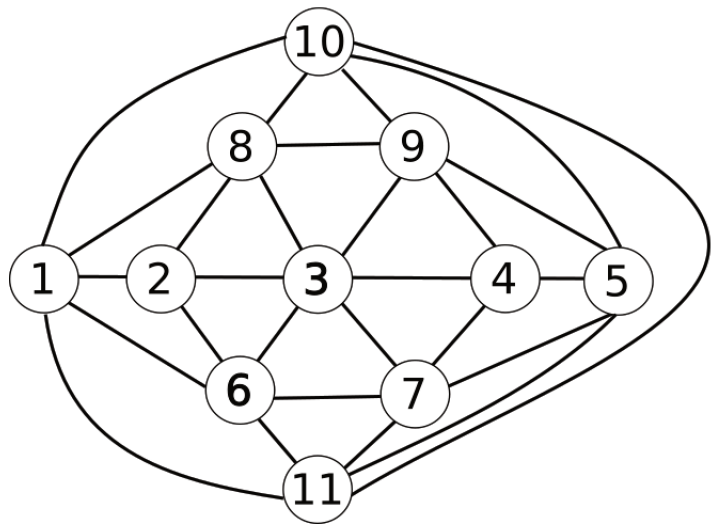
Os grafos são estudados em problemas relacionados a SP desde o final da década de 60. Alguns dos primeiros algoritmos desenvolvidos foram os de Miller (1970) e Grason (1971). O algoritmo de Miller cria grafos através de uma matriz de adjacência que é inserida como dado de entrada pelo usuário (figura 14). A matriz é composta dos números 0 e 1, sendo que alternativa 0 corresponde a não relação entre ambientes e a 1 determina que existe relação. Nesse algoritmo as relações são sempre bidirecionais. Essa matriz é reorganizada de acordo com uma sub-rotina que aproxima as relações da sua diagonal, maximizando a proximidade entre os ambientes que se inter-relacionam (figura 16). Então o algoritmo identifica aglomerados (*clusters*) de relações e o mapeia em um grafo (figura 15).

Figura 12 – Grafo planar(A) e Grafo não planar(B)



Fonte: Figura produzida pelo autor

Figura 13 – Grafo planar máximo



Fonte: Figura produzida pelo autor

Figura 14 – Matriz de adjacência de Miller

		1	2	3	4	5	6	7
Sala do diretor	1	1			1	1		
Secretaria	2		1				1	1
Livraria	3		1	1			1	
Sala da secretaria	4	1			1	1		1
Sala de conferência	5	1				1		
Diretoria	6		1	1			1	
Lobby	7		1		1			1

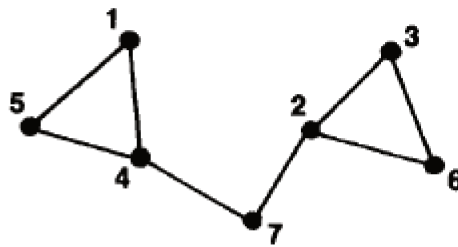
Fonte: Adaptado de "Computer-aided space planning" (MILLER, 1970)

Figura 15 – Matriz de adjacência revisada para maximizar diagonal de Miller

		1	5	4	7	2	3	6
Sala do diretor	1	1	1	1				
Sala de conferência	5	1	1	1				
Sala da secretaria	4	1	1	1	1			
Lobby	7			1	1	1		
Secretaria	2				1	1	1	1
Livraria	3					1	1	1
Diretoria	6					1	1	1

Fonte: Adaptado de "*Computer-aided space planning*" (MILLER, 1970)

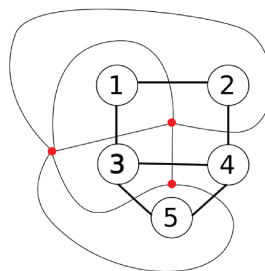
Figura 16 – Matriz de adjacência revisada mapeada em um grafo



Fonte: "*Computer-aided space planning*" (MILLER, 1970)

O algoritmo de Grason foi denominado GRAMPA (*GR*aph *MA*nipulating *PA*c-kage). O método é baseado em um grafo dual. Um grafo dual é um grafo derivado de um primeiro grafo X, em que para cada face de X (face é uma região obtida por arestas que definem um plano), há um vértice de X' e estes vértices se interligam através de arestas por regiões adjacentes de X. O GRAMPA foi desenvolvido com uma pseudo gramática chamada PGG (*Planar Graph Grammar*), que especifica os grafos.

Figura 17 – Exemplo de grafo dual



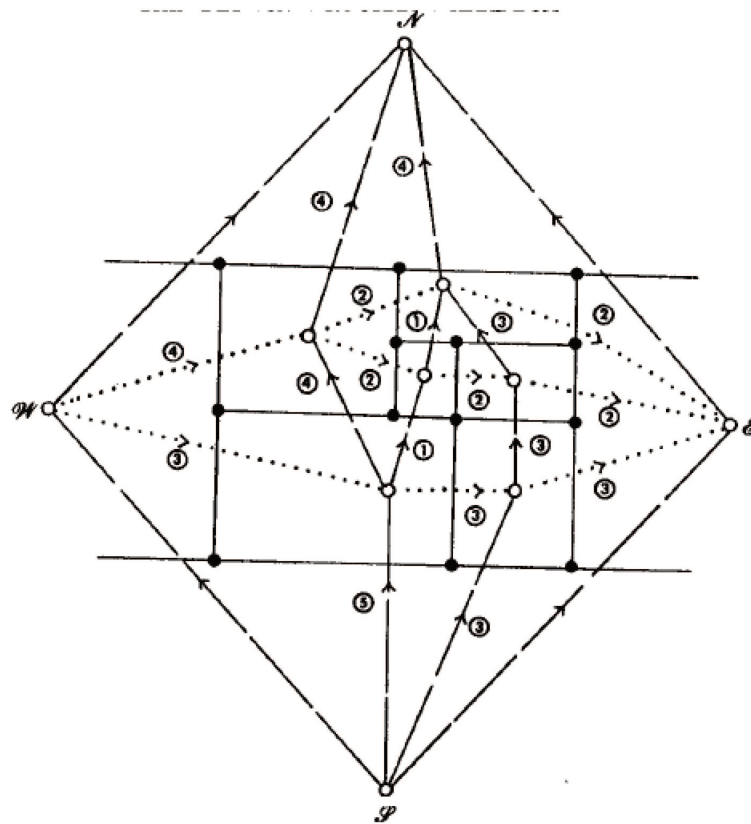
Fonte: "*Computer-aided space planning*" (MILLER, 1970)

Os dados de entrada do GRAMPA são: o programa de necessidades arquitetônico com as dimensões físicas especificadas, e as relações de adjacência correspondentes aos

ambientes. Então é gerado um grafo planar onde as arestas são segmentos de parede e os nós são as quinas. O *dual graph* é criado a partir de um nó colocado no centro de cada ambiente e arestas que interligam os espaços adjacentes entre esses nós (figuras 17 e 18). Por convenção do programa, segmentos de aresta que passam no sentido norte-sul são traçados como (.....) e segmentos que passam no sentido leste-oeste são traçados (—). Cada aresta recebe o valor da dimensão do segmento de parede que atravessa. Depois de criado o *dual graph* uma sub-rotina avalia: (1) se o grafo é planar (2) se os nós formam uma região (figura 19), (3) a orientação das arestas para determinar se são norte-sul ou leste-oeste.

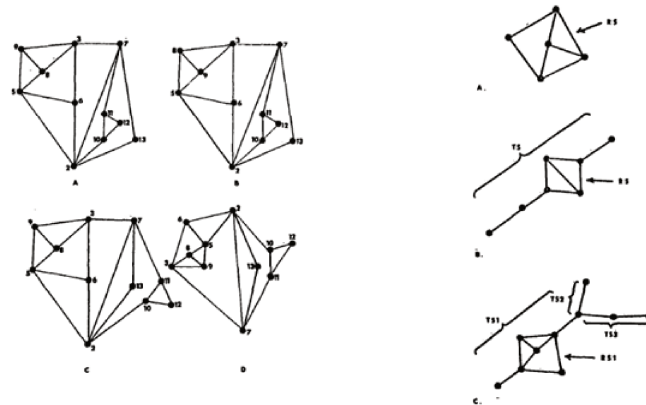
Para gerar alternativas geométricas para os grafos é usada a PGG. Essa pseudo gramática divide grafo em subgrafos que podem ser tratados como uma entidade separada. Os subgrafos são livres para reconfigurar sua posição, a fim de criar diferentes configurações espaciais para os grafos (figura 7). Depois de aplicada a gramática (PGG), uma série de subrotinas avaliam planaridade, requerimentos de adjacência, formação de região através dos nós e requerimentos de dimensão. O objetivo do GRAMPA é gerar diferentes configurações de layout com uma mesma matriz de adjacência (figura 8).

Figura 18 – Grafo dual de Grason



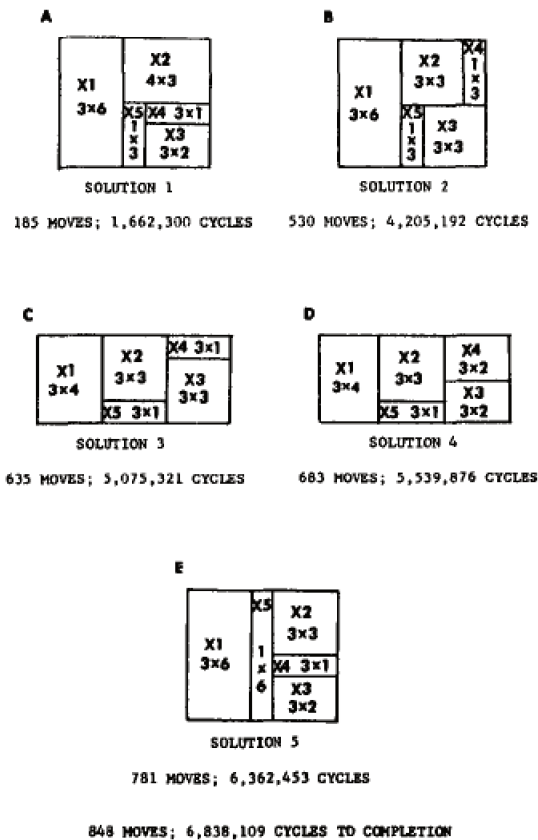
Fonte: Figura produzida pelo autor

Figura 19 – Diferentes configurações espaciais dos grafos à esquerda e subgrafos a direita



Fonte: “An approach to computerized space planning using graph theory” (GRASON, 1971)

Figura 20 – Diferentes soluções de layout para um mesmo problema GRAMPA



Fonte: “An approach to computerized space planning using graph theory” (GRASON, 1971)

Outros algoritmos buscam maximizar a soma de benefícios através de um valor atribuído à aresta. Esse valor, denominado peso, representa o grau de interação entre os

elementos correspondentes (vértices). Se as interações entre ambientes são uma matriz de fluxo material, podemos usar os números dos fluxos de materiais como pesos.

2.2.3 *Improvement Algorithms*

Os *Improvement Algorithms* são algoritmos que se tornam mais eficazes a cada iteração. Eles têm sido aplicados com sucesso em problemas de difícil solução, como os de cálculo combinatório (HERAGU, 2008). Liggett (2000) exemplifica um modelo simples de *Improvement Algorithm*: dada uma solução inicial, o procedimento consiste em avaliar sistematicamente as possibilidades de intercambio entre pares de atividades, realizando a troca se o procedimento aprimorar o valor especificado (LIGGETT, 2000).

O CRAFT é um software desenvolvido por Armour e Buffa em 1963, popular nas pesquisas em *space planning* (KUSIAK; HERAGU, 1987). O software opera determinando o custo inicial do layout, então avalia as possíveis mudanças de localização entre espaços pareados adjacentes ou que tenham a mesma dimensão. As mudanças de localização dos espaços continuam até que o custo estimado do layout seja reduzido ao máximo.

O professor Paul Jesen da Universidade do Texas desenvolveu uma versão do software *Computerized Relative Allocation of Facilities Technique* (CRAFT) implementada em Excel (disponível para download em <http://www.me.utexas.edu/~jensen/ORMM/omie/computation/unit/lay_add/lay_add.html>). Nesse software, a criação do layout se dá da seguinte forma: definição do nome do projeto, do número de espaços, do número de pontos fixos e da medida de distância. Quando opção *Make Random Problem* é marcada, os fluxos de materiais entre os ambientes são definidos randomicamente.

O usuário deve preencher o comprimento, a altura e as distâncias que serão convertidas em células através do fator de escala especificado. O CRAFT tem um limite de dimensão de 50 células de altura por 100 células de comprimento. Quando o layout excede as dimensões limite o fator de escala deve ser ajustado acima de 1, convertendo a medida de distância para medida de célula. Um fator de escala maior que 1 reduz o tamanho do layout e resulta em soluções mais rápidas.

Os dados dos ambientes são definidos na linha 16 da planilha. Na coluna B estão os nomes, na coluna C é definido se o ambiente é fixo (F) ou variável (V), isso significa que a localização ou a sequencia podem variar de local na busca de uma solução. Na coluna D estão definidas as áreas em metros quadrados e na coluna E as dimensões dos ambientes em células.

A planilha *Flow Matrix* é onde o usuário deve definir as relações de fluxo de materiais entre os ambientes. Abaixo dessa planilha existe uma outra que determina o custo de transporte entre os espaços. Por padrão, as relações entre os ambientes são definidas como 1, isso significa que o custo do fluxo dos materiais entre os ambientes é

o mesmo. Entretanto, esse valor pode ser alterado para refletir uma situação em que o custo de transporte entre ambientes é dificultado, como, por exemplo, na existência de um deslocamento vertical.

Quando todas as variáveis estão definidas utiliza-se a opção *Define Facility*, que apresenta uma nova caixa de diálogo contendo as opções de *Solution Method*, *Distance Measure*, dentre outras. *Solution Method* define o tipo de otimização que será feita.

Distance Measure define o método de medida de distância que será usado: retilíneo ou euclidiano. O método retilíneo, também conhecido como *Manhattan* é obtido pela soma das distâncias verticais e horizontais. Por exemplo, dado um vetor A(4.0, 6.0) e um vetor B(8.0,6.0), temos ($|4.0-8.0|+|6.0-6.0| = 4 + 0 = 4$). No caso do método *Euclidean*, também conhecido como *Straight*, o cálculo é feito pela distancia geométrica, então temos $\sqrt{(4.0 - 8.0)^2 + (6.0 - 6.0)^2} = 4$ (KADO, 1996). Ao selecionar a opção OK após a definição dos parâmetros, o cálculo inicial do *layout* é feito.

Esse é o primeiro *layout* gerado, sem nenhuma interação para troca de localização de ambientes. A troca de pares de ambientes é feita através da opção *Solve*. Através dessa opção existem duas possibilidades de iteração: a primeira exhibe as interações uma a uma, apresentando a cada passo a redução do custo de *layout*, possibilitando o usuário aceitar ou não o intercambio de ambientes. Na segunda possibilidade, depois que todas as operações possíveis de trocas foram realizadas, é apresentado o resultado final com menor custo, sem que haja a interação do usuário.

Nesta etapa são demonstradas as mudanças ocorridas no custo de transporte, que foram obtidas na etapa anterior. Por fim é possível visualizar os fluxos de materiais entre os ambientes, através da opção *Show Flows*.

2.3 Considerações sobre Space Planning

Problemas de SP são considerados de difícil resolução e diferentemente dos problemas de decisão, onde se espera uma resposta de sim ou não, os algoritmos destinados a resolver problemas de SP não são determinísticos, ou seja, existem diferentes comportamentos e respostas para um mesmo dado de entrada. Os algoritmos heurísticos, por buscarem uma resposta viável, são mais utilizados para esse tipo de problema (HERAGU, 2008)

Dentre os métodos heurísticos, este trabalho estudará os algoritmos de space planning com abordagem evolutiva, que fazem parte da classe de *improvement algorithms* (LIGGETT, 2000), pois as soluções se aperfeiçoam no decorrer das interações e o objetivo está centrado na melhor solução, dentro de um tempo computacional aceitável. O capítulo 4 será todo destinado a esse tema.

O próximo capítulo foi destinado aos algoritmos evolutivos do ponto de vista de suas características computacionais, além de apresentar alguns aspectos teóricos e experimentos arquitetônicos que utilizaram a abordagem evolutiva como processo de projeto.

Parte II

Os Algoritmos Evolutivos

3 Algoritmos Evolutivos

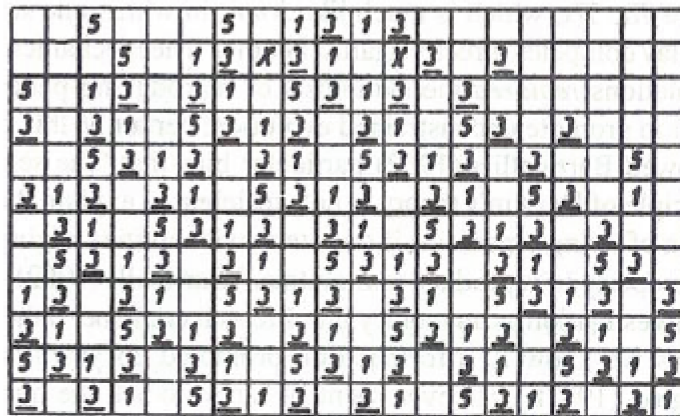
Algoritmos evolutivos (AE) são parte de um campo de pesquisa denominado Computação Evolutiva, um ramo específico da inteligência artificial (IA), que aborda um conjunto de métodos de soluções de problemas baseados na evolução natural, sendo esses métodos implementados computacionalmente (GABRIEL; DELBEM, 2008).

No início da década de 30 foram desenvolvidos os primeiros estudos de algoritmos baseados em sistemas evolutivos naturais. Entretanto, somente a partir do início da década de 60, com a queda do preço dos computadores e o aumento de sua disponibilidade, esses estudos se intensificaram (JONG, 2006).

O primeiro trabalho publicado em computação evolutiva foi feito por Barricelli em 1953, quando trabalhava no projeto de John Von Neumann para um computador de circuitos de alta velocidade, no Institute for Advanced Study em Princeton (FOGEL, 2006). Os experimentos de Barricelli eram baseados em princípios de IA, sendo que um destes experimentos foi a criação de um grid em que números eram dispostos aleatoriamente nas células do topo deste grid, então os números trocavam de célula de acordo com regras locais de interação (FOGEL, 2006).

A figura 21 ilustra o trabalho de Barricelli, através de números que se propagam no tempo em um grid de uma série de 20 células. Os números que preenchem as células são -3, 1, 5 e o 0 que representa um espaço vazio na célula. O processo de propagação se inicia no topo do grid e, a partir da quarta geração, a sequência (5, -3, 1, -3, 0, -3, 1) aparece pela primeira vez e persiste no tempo através de gerações (FOGEL, 2006).

Figura 21 – Grid com uma serie de números se movendo baseados em nas regras de Barricelli



Fonte: "Foundations of evolutionary computation" (FOGEL, 2006)

Os estudos de computação evolutiva iniciaram seu desenvolvimento na década de

60 de modo paralelo e independente. Esses estudos eram: Estratégias Evolutivas, Programação Evolutiva e Algoritmos Genéticos (GABRIEL; DELBEM, 2008).

Em 1965 na universidade técnica de Berlin, Rechenberg e Schwefel começaram a desenvolver a ideia de um algoritmo chamado “*evolution strategies*” (Estratégias Evolutivas), que explorava processos evolutivos que poderiam ser usados para resolver problemas de difícil otimização de funções de números reais (JONG, 2006).

Durante o mesmo período na UCLA (*University of California, Los Angeles*), Lawrence Fogel explorava conceitos de IA para gerar agentes inteligentes que representavam uma máquina de estados finitos, através desses estudos foi desenvolvida uma plataforma chamada “*Evolutionary Programming*” (Programação Evolutiva), bem sucedida em evoluir bons agentes de uma máquina finita de passos. Essa plataforma continua a ser atualizada atualmente, sendo aplicada a uma série de problemas que vão além do propósito original (JONG, 2006).

Foi na universidade de Michigan que John Henry Holland desenvolveu o conceito de algoritmo evolutivo, através do entendimento de que os processos evolutivos deveriam ser capazes de lidar com as mudanças e incertezas do ambiente. A ênfase de seu trabalho estava em criar um sistema que fosse capaz de se auto adaptar no tempo, como uma função de feedback obtida através da interação com o ambiente em que opera (JONG, 2006).

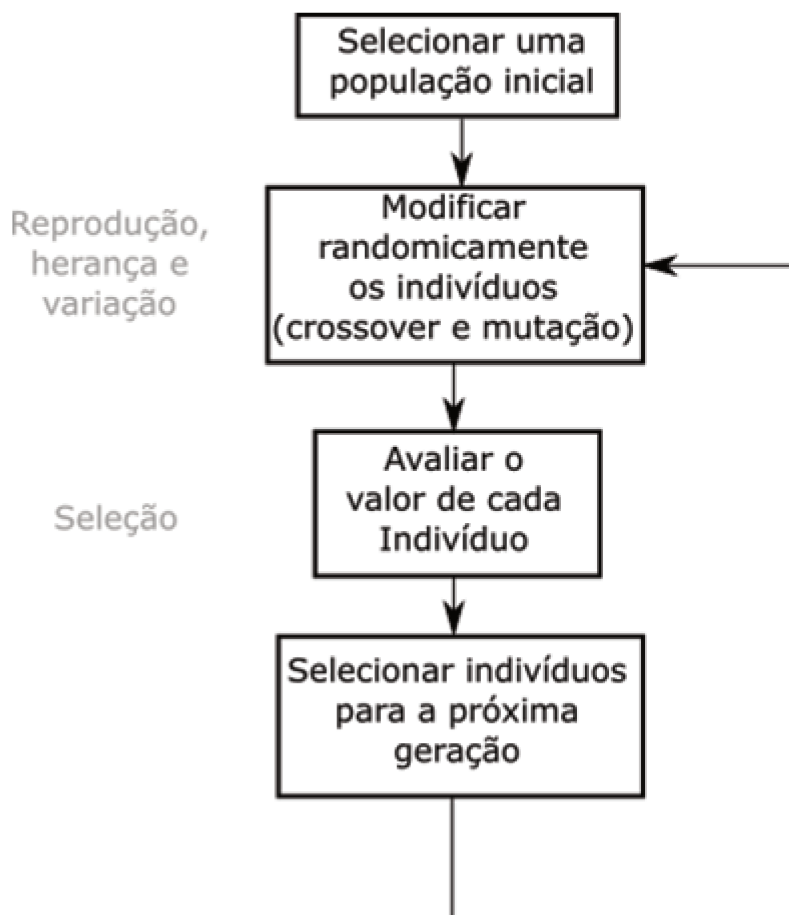
Existem quatro condições essenciais para que haja evolução baseando-se na seleção natural da teoria de Darwin (BANZHAF *et al.*, 1998):

1. Hereditariedade na reprodução
2. Reprodução de indivíduos em uma população
3. Variações que afetam a probabilidade da sobrevivência dos indivíduos
4. Recursos finitos para causar competição (seleção)

Hereditariedade é o processo pelo qual os filhos recebem as propriedades de seus pais, ou seja, se esse ser vivo viver o suficiente para se reproduzir, suas características podem ser transmitidas para os seus filhos na próxima geração de seres da espécie (SHIFFMAN *et al.*, 2012). **Variação** compreende as distintas características que compõem os indivíduos de uma população (SHIFFMAN *et al.*, 2012). **Seleção** é o mecanismo pelo qual se regula quais indivíduos devem ter a oportunidade de se reproduzir e passar informações genéticas com suas características para a próxima geração. Tipicamente é o mecanismo chamado de “sobrevivência do mais apto”. Esse mecanismo em algoritmos é definido por uma função denominada função objetivo ou *fitness function* (SHIFFMAN *et al.*, 2012).

Essas características comuns a todos os algoritmos evolutivos podem ser ilustradas através do fluxograma desenvolvido por Fogel (2006):

Figura 22 – Fluxo de funcionamento dos algoritmos evolutivos



Fonte: Adaptado de "*Foundations of evolutionary computation*" (FOGEL, 2006)

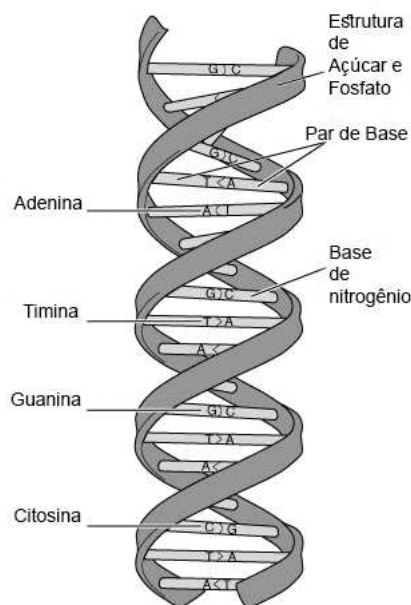
Entretanto, diferentemente dos sistemas naturais, Bentley (1999) afirma que algoritmos evolutivos dependem ainda de mais 3 características importantes: **inicialização**, **avaliação** e **conclusão**.

Inicialização determina a dimensão da população através de soluções com valores randômicos. **Avaliação** em AE é o procedimento responsável por guiar a evolução através das melhores soluções. No caso dos algoritmos evolutivos, não há um ambiente natural como a natureza que meça a sobrevivência e as qualidades das soluções para testá-las; em um ambiente computacional as avaliações das soluções são simuladas e calculadas. A **conclusão** em um AE é estabelecida pela convergência da população para uma solução determinada através de uma resposta viável para o dado problema, ou quando é estabelecido um número predeterminado de gerações para o termino das iterações. No caso da evolução natural somente a extinção garante o termino do processo evolutivo (BENTLEY, 1999)

3.1 Da codificação da vida à codificação de algoritmos

O DNA (Ácido Desoxirribonucleico), apresentado ao mundo por Watson e Crick em 1953, é o responsável pela codificação de todas as informações genéticas necessárias para a constituição dos seres vivos. Sua estrutura é formada por uma espiral dupla composta por duas fitas que carregam moléculas alternadas de açúcar (desoxirribose) e fosfato, em que a cada molécula de açúcar está associada uma destas quatro bases: Adenina (A), Timina (T), Citosina (C) ou Guanina (G) (MARTINO, 2015). As duas fitas são unidas através de bases compostas por pares de Adenina com Timina e também Citosina com Guanina (figura 23). Através da sequência das bases ao longo de toda a fita serão fornecidas as instruções genéticas (genótipo) necessárias para a formação das proteínas, e quando compostas com outros lipídios, juntamente com outras moléculas, formam os fenótipos dos seres vivos (MARTINO, 2015 apud MAYR, 2009). Genótipo em biologia é o conjunto de genes de um indivíduo que não são modificados naturalmente, ou seja, às características internas, à constituição genética, o conjunto de cromossomos ou sequência de genes herdado dos pais, enquanto o conceito de fenótipo é o resultado da interação do meio com o conjunto de genes do indivíduo, sendo as suas características externas, morfológicas, fisiológicas e comportamentais.

Figura 23 – Representação do DNA

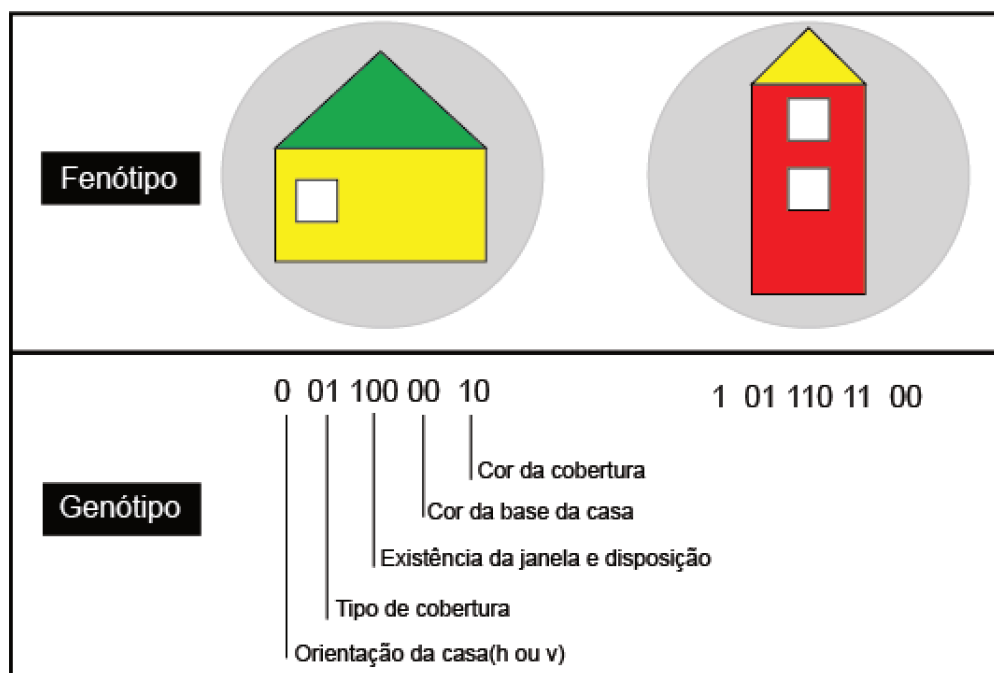


Fonte: Adaptado de Adaptado de Wikimedia “<<https://upload.wikimedia.org/wikipedia/commons/b/b8/DNA-structure-and-bases.png>>”

O DNA compõe os cromossomos dos núcleos das células de todos os seres vivos. O ser humano possui 46 pares de cromossomos, sendo 23 de origem paterna e os outros 23 de origem materna. Como nos seres vivos que compõem o ambiente natural, os candidatos

a solução em um AE também são codificados através de um cromossomo. Em princípios gerais, o cromossomo representa uma série de parâmetros ou valores de uma candidata à solução e algumas vezes a própria solução. Existem distintos métodos para se codificar e representar um cromossomo em um AE, um deles é por meio de um sistema binário (0101010) (figura 24), em que um ou mais valores de bit representam um parâmetro de uma solução candidata.

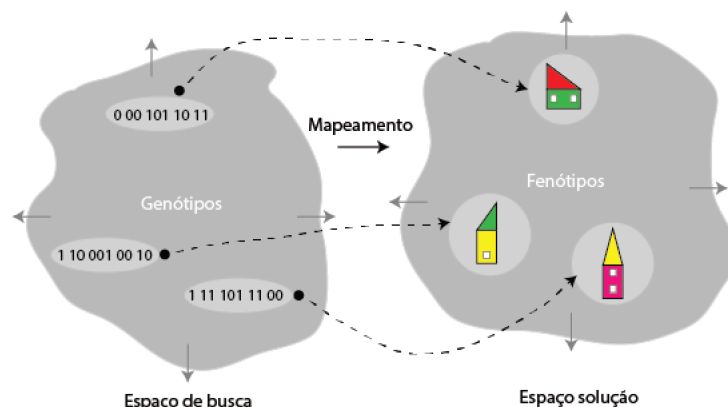
Figura 24 – Representação de uma casa através dos fenótipos e genótipos



Elaborado pelo autor

O principal objetivo dos algoritmos evolutivos consiste em buscar soluções para o dado problema (BENTLEY, 1999). O espaço que compõe todas as possíveis soluções de um problema é chamado de “espaço de busca” (*search space*)(figura 25). Nesse espaço podem existir, teoricamente, infinitas possibilidades de se localar um ponto e cada possível configuração representa uma solução candidata.

Figura 25 – Espaço de busca e espaço solução



Fonte: Adaptado de “*Evolutionary Design by Computers* “(BENTLEY, 1999)

No **espaço de busca** a população de um AE é inicializada. Essa população é composta por elementos codificados através cromossomos. Os indivíduos são avaliados e selecionados de acordo com a interação entre os métodos de seleção e a função objetivo, determinando quais indivíduos passarão para a próxima geração.

A função objetivo é formulada de acordo com o problema que se deseja resolver. Os indivíduos mais aptos dessa população se **reproduzem** gerando descendentes que carregam parte dos cromossomos de cada um dos pais. Esse procedimento recebe o nome de **recombinação**. Uma pequena parcela dos indivíduos da população tem parte de seu cromossomo modificado randomicamente (**mutação**).

Após esses procedimentos, o passo de **conclusão** verifica, dentre os melhores indivíduos da última geração, se existem indivíduos bons o suficiente para a solução do problema. Se algum indivíduo atingir o objetivo (no caso de uma avaliação em que se busca um número exato), então o processo é terminado. Caso contrário uma nova geração é criada e o processo é repetido por um dado número de gerações ou até que uma solução aceitável seja obtida.

Como no ambiente natural, em que há diferença entre o genótipo (informações codificadas que compõem os seres vivos) e fenótipo (características do indivíduo), as soluções de AE são codificadas através de um cromossomo, em que cada gene representa um parâmetro e que juntos formalizam uma solução.

3.2 Os métodos evolutivos

Apesar dos processos dos algoritmos evolutivos serem muito semelhantes, compartilhando algumas etapas em comum, cada método evolutivo tem características singulares. Alguns autores (MICHALEWICZ, 2013; ZUBEN, 2000) preferem tratar os métodos evolutivos como "algoritmos evolutivos", por conta da estrutura dos algoritmos ser muito

semelhante. Outros (JONG *et al.*, 1997; BENTLEY, 1999) preferem a categorização por método empregado. Neste trabalho discutiremos sobre alguns dos principais métodos, bem como as etapas de funcionamento de seu processo, mas é bom evidenciar que os métodos de algoritmos evolutivos desenvolvidos atualmente tendem a mesclar técnicas dificultando assim a classificação do método empregado de maneira precisa.

3.2.1 Algoritmos Genéticos

Um algoritmo genético (AG) é um algoritmo estocástico modelado através de métodos de busca baseados em alguns fenômenos naturais, como por exemplo hereditariedade genética e o conceito de luta e sobrevivência de Darwin (MICHALEWICZ, 2013).

O primeiro AG desenvolvido por Holland utilizava representação cromossômica binária com tamanho fixo, ou seja, os indivíduos de uma população eram representados através de um cromossomo de *bits* compostos por números 0 e 1, sendo que todos os indivíduos da população possuíam um tamanho fixo (BANZHAF *et al.*, 1998).

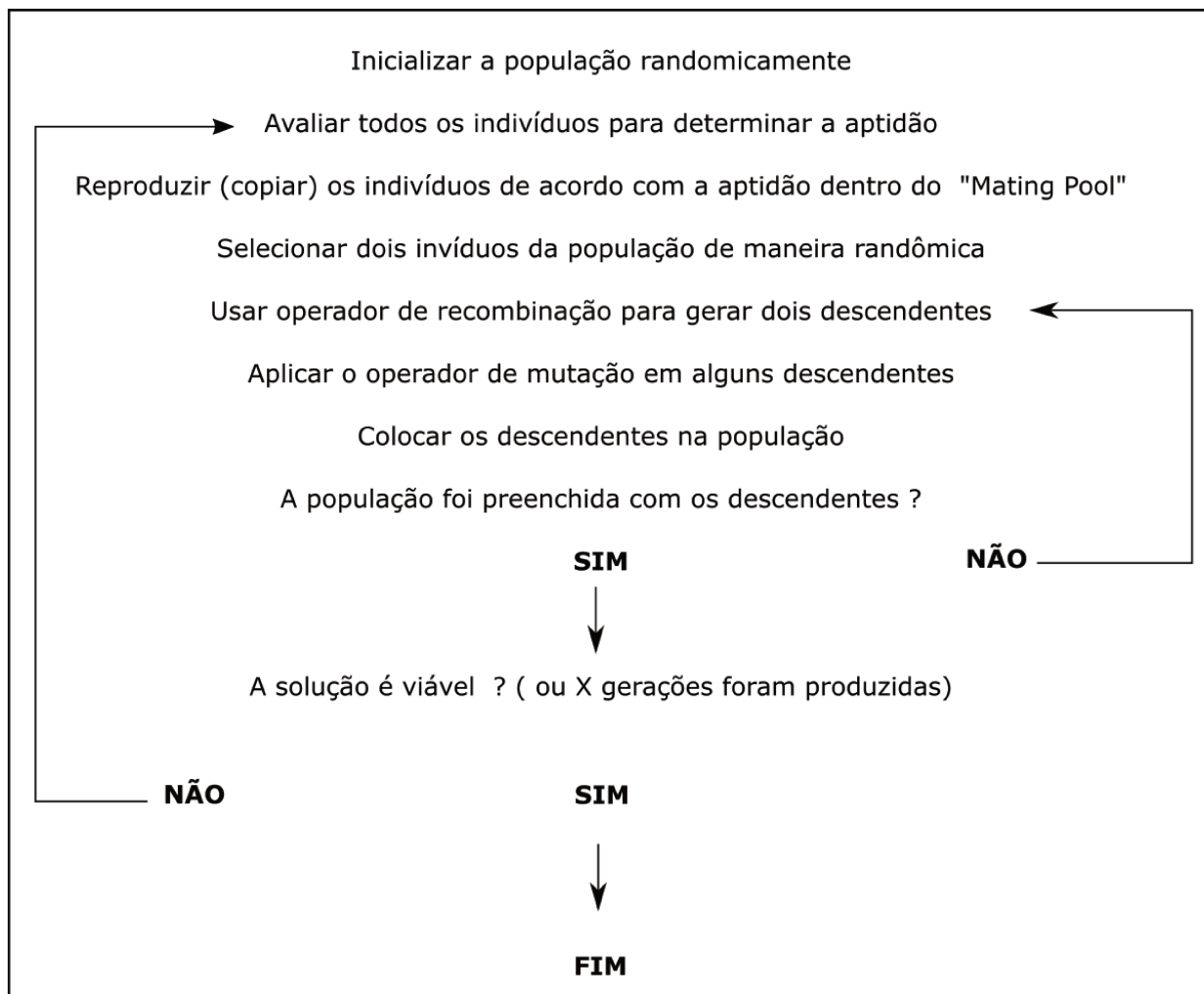
Tabela 2 – Tamanho fixo do cromossomo

População	Cromossomos	Quantidade de genes
Indivíduo 1	010010001001	12
Indivíduo 2	101101010010	12
Indivíduo 3	100100111101	12

Elaborado pelo autor

Abaixo ilustraremos o funcionamento de um AG através do fluxograma desenvolvido por Bentley (1999):

Figura 26 – Fluxograma do funcionamento de um algoritmo genético.

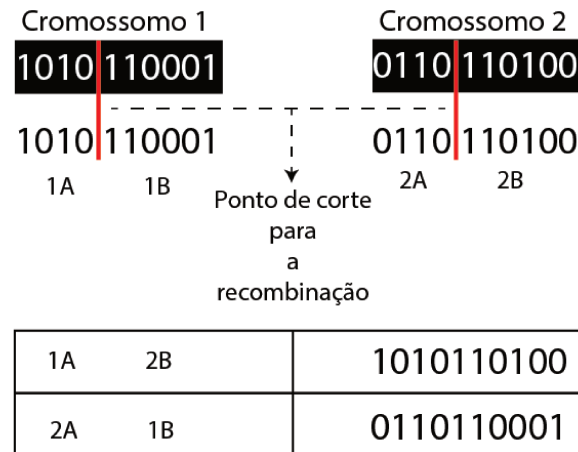


Fonte: Adaptado de "*Evolutionary Design by Computers*"(BENTLEY, 1999)

O genótipo de cada indivíduo da população é inicializado por uma combinação randômica. O algoritmo então é iniciado e a representação do fenótipo correspondente é avaliada de acordo com a função objetivo determinada para o problema. Cada fenótipo (solução) recebe um valor de aptidão correspondente à sua aptidão. Esse valor é utilizado para determinar quantas cópias de cada indivíduo serão alocadas na memória temporária do algoritmo, denominada "*mating pool*", (quanto maior o valor de aptidão, mais cópias são feitas de um indivíduo). Dois indivíduos são selecionados randomicamente do "*mating pool*". A descendência é feita através de operadores de **recombinação** que randomicamente alocam genes dos pais para os genes da descendência. A **mutação** ocasionalmente ocorre com uma baixa probabilidade na descendência (BENTLEY, 1999).

Os algoritmos genéticos enfatizam como principal operador de diversidade o processo de recombinação, sendo a mutação apenas um fato secundário, com uma baixa taxa de probabilidade de acontecimento (MARTINO, 2015)

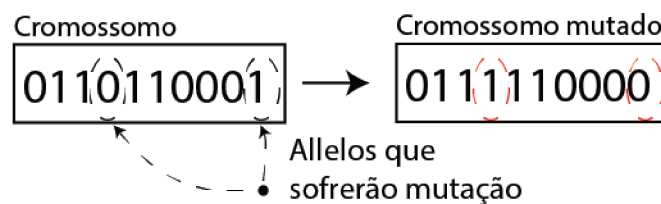
Figura 27 – Operador de recombinação



Fonte: Elaborado pelo autor

A descendência é gerada até que a população seja novamente preenchida (todos os pais são descartados). O processo é repetido por um número determinado de gerações ou até que uma solução viável ou ótima seja obtida (BENTLEY, 1999).

Figura 28 – Operador de mutação



Fonte: Elaborado pelo autor

3.2.1.1 Métodos de seleção

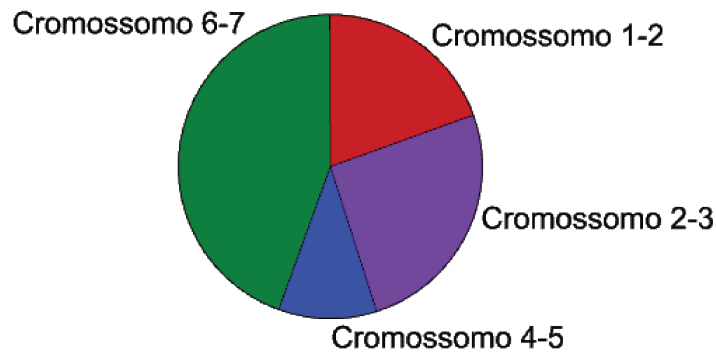
Existem diferentes métodos de seleção de indivíduos de uma população de GA. São esses métodos que determinarão quais indivíduos terão a oportunidade de se reproduzir, e esta seleção está diretamente relacionada com a função objetivo (POBLETE, 2007). Cada método de seleção é personalizado para enfrentar distintas características de problemas, sendo que alguns prezam por diminuir o custo computacional, e outros por selecionar indivíduos mais aptos, entre outros (POBLETE, 2007).

Os principais métodos de seleção de um AG são:

Seleção de Roleta (*Roulette Wheel*)

A Seleção de Roleta é uma das técnicas de seleção mais usadas em AG (POBLETE, 2007). Esse tipo de seleção é proporcional à aptidão de cada indivíduo, sendo-lhe destinada uma porcentagem da roleta (figura 29)

Figura 29 – Seleção de roleta



Fonte:Elaborado pelo autor

Cada indivíduo da população se enquadra em uma determinada parte da roleta, proporcional aos seus valores de aptidão (MITCHELL, 1998). A soma total das porcentagens de indivíduos corresponde ao valor total da roleta. Através desse método, os melhores indivíduos recebem uma porção maior da roleta e têm mais chances de ser selecionados para a reprodução.

O método de seleção desenvolvido por Mitchell (1998) segue os passos descritos no pseudocódigo 3.2.1.1.

Pseudocódigo 3.1 PSEUDO-CÓDIGO SELEÇÃO DE ROLETA

T = soma dos valores de aptidão de todos os indivíduos da população
repeat
 Selecionar n indivíduos
 r = valor aleatório entre 0 e T , percorra sequencialmente os indivíduos da população, acumulando em S o valor de aptidão dos indivíduos já percorridos
until $S \geq r$
 Selecione o indivíduo corrente

Esse método é eficiente somente quando aplicado a pequenas populações, e usualmente a quantidade de indivíduos que contera a roleta não tem um tamanho fixo, sendo necessário definir a quantidade de vezes que o melhor indivíduo será selecionado. Espera-se com esse procedimento que haja redução nos custos computacionais, bem como controle na porcentagem de seleção de soluções piores (POBLETE, 2007).

Seleção por Torneio

O método de seleção por torneio consiste em escolher randomicamente dois indivíduos da população e então selecionar o indivíduo mais apto, de acordo com a função objetivo. O processo continua até que o número de indivíduos seja igual ao tamanho da

população (POBLETE, 2007). É através da quantidade de indivíduos no torneio que se define a pressão seletiva.

Existem dois tipos de seleção por torneio. O primeiro método é chamado de **determinístico** e seleciona o melhor dentre os dois indivíduos, garantindo um grau de elitismo, porém produz uma convergência prematura da população. O segundo método é o **probabilístico** que é definido através da geração de um número r aleatório entre 0 e 1. Sendo esse r maior que um determinado parâmetro k , se $r < k$, o indivíduo mais apto é selecionado, caso contrário o escolhido é o indivíduo menos apto (MITCHELL, 1998).

Seleção Elitista

A seleção elitista, desenvolvida por Kenneth De Jong em 1975 (MITCHELL, 1998), corresponde a um grupo de métodos que força o AG a manter um determinado número de melhores soluções para a próxima geração. Esse método garante que os melhores indivíduos reproduzam e transmitam seus genes para a próxima geração. Muitos pesquisadores acreditam que o método elitista melhora significativamente o desempenho dos AGs (MITCHELL, 1998), enquanto outros defendem que esse método deve ser mesclado a outros, caso contrário leva à convergência prematura da solução do problema, o que significa que a solução converge rapidamente em um resultado subótimo, pois os descendentes não conseguem superar os resultados dos pais e uma das causas para isso é a pouca variação genética da população (POBLETE, 2007). Um ótimo local em problemas de otimização é uma solução que é ótima dentre um conjunto de soluções vizinhas, diferentemente de uma solução ótima global, em que a solução é ideal entre todas as soluções possíveis.

Seleção por Ranqueamento

Um dos principais propósitos do método de Ranque é evitar a convergência prematura da solução. Na solução proposta por Baker, em 1985, os indivíduos de uma população são ranqueados de acordo com sua aptidão, e o valor esperado de cada indivíduo depende de seu ranque mais do que do valor de *fitness* absoluto (MITCHELL, 1998). A vantagem desse método é que ele evita que os indivíduos mais aptos ganhem domínio sobre os outros (POBLETE, 2007) e uma desvantagem é que, por vezes, pode ser necessário saber se um indivíduo da população é muito mais apto que um outro próximo (MITCHELL, 1998).

O método proposto por Baker segue os seguintes passos: Cada indivíduo da população é ranqueado em ordem crescente de aptidão, de 1 a N . O usuário determina os valores máximos esperados de cada indivíduo através deste valor N . O valor esperado $ExpVal$ de cada indivíduo i de uma população em um tempo t é dado por (MITCHELL, 1998):

$$ExpVal(i, t) = Min + (Max - Min) \frac{rank(i, t) - 1}{N - 1}$$

3.2.1.2 Os métodos de recombinação (*Crossover*)

O operador de recombinação é o principal mecanismo de reprodução e variação em AG (GABRIEL; DELBEM, 2008 apud GOLDBERG; HOLLAND, 1988). Esse mecanismo é responsável pela obtenção de novos indivíduos pela troca ou combinação de alelos de dois ou mais indivíduos (GABRIEL; DELBEM, 2008). Os alelos são os valores que um gene pode assumir, por exemplo, um gene que representa o material de uma parede pode ter alelos como : tijolos, blocos, taipa, etc. Parte de um indivíduo é trocada por uma parte correspondente de outro indivíduo e como resultado temos um descendente que combina características de cada um dos pais. Existem distintos métodos de recombinação de cromossomos. Alguns deles são apresentados a seguir:

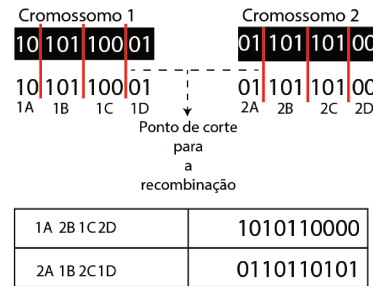
Recombinação ponto a ponto

A recombinação ponto-a-ponto é feita através da seleção de um ponto de corte aleatório no cromossomo, dividindo-o em duas partes, esquerda e direita. Cada descendente é definido por uma parte esquerda e uma parte direita do ponto de corte (GABRIEL; DELBEM, 2008)(figura 27).

Recombinação n pontos

Cada cromossomo é subdividido em n partições que serão recombinadas para criar a descendência (GABRIEL; DELBEM, 2008). A figura 30 ilustra um exemplo com 3 pontos de corte.

Figura 30 – Recombinação através de 3 pontos de corte



Fonte: Elaborado pelo autor

Recombinação Uniforme

Na recombinação uniforme cada gene é considerado de forma independente, então é criada uma lista de variáveis aleatórias de distribuição uniforme entre $[0,1]$ atribuindo um resultado para cada indivíduo. Caso o valor da variável seja inferior a 0,5 o gene será oriundo do pai 1, caso contrário será do pai 2 (GABRIEL; DELBEM, 2008).

Tabela 3 – Recombinação uniforme

Cromossomo pai 1	0	1	1	0	1	0	1
Cromossomo pai 2	1	1	0	1	0	1	1
Valor sorteado 3	0.5	0.7	0.1	0.8	0.5	0.2	0.6
Filho 1	0	1	0	0	1	1	1
Valor sorteado 2	0.4	0.3	0.9	0.8	0.7	0	0.1
Filho 2	1	1	1	0	1	1	1

Fonte: Elaborado pelo autor

3.2.1.3 Mutação

É tarefa do operador de mutação modificar aleatoriamente um ou mais genes de um cromossomo (GABRIEL; DELBEM, 2008). Esse processo pode ocorrer em uma pequena taxa da população, em que os descendentes tenham herdado as características dos pais. Isso é feito para ampliar a busca através do espaço de soluções e para que haja uma maior probabilidade de melhoria na aptidão da próxima geração que será proveniente do aumento de variação (POBLETE, 2007). A **taxa de mutação** de uma população é usualmente pequena, em decorrência da possibilidade de se gerar um indivíduo potencialmente pior que o original (POBLETE, 2007).

Considerando que a representação do cromossomo seja binária, o operador de mutação simplesmente elege um ou mais pontos aleatórios no cromossomo e o troca pelo valor inverso, por exemplo, se o alelo selecionado é 1 será modificado para 0, caso a representação não seja binária, o alelo é substituído randomicamente.

O operador de recombinação é responsável por uma busca rápida e eficiente no espaço de solução, enquanto o operador de mutação é encarregado de explorar de modo mais amplo todo o espaço solução, contribuindo desse modo para uma não convergência prematura (POBLETE, 2007).

3.2.2 Estratégias Evolutivas

Os primeiros algoritmos de estratégias evolutivas (EE) foram desenvolvidos em 1960 e seus objetivos estavam ligados à otimização de parâmetros (GABRIEL; DELBEM, 2008) e inicialmente tratavam de problemas de otimização em mecânica (FOGEL, 2006). Rechenberg (1965) e Schwefel (1965) desenvolveram um algoritmo denominado *(1+1) EE* ou *two membered ES*. Esse algoritmo gerava através de um único pai um descendente e ambos competiam pela sobrevivência (GABRIEL; DELBEM, 2008).

Esse algoritmo segue os seguintes passos:

A solução filho é gerada randomicamente por mutação dos parâmetros do pai. Os cromossomos são compostos por dois *arrays* (duas listas com valores), um com os dos cromossomos e outro composto pelo desvio padrão desses valores (GABRIEL; DELBEM, 2008). O desvio padrão é a média dos valores absolutos do desvio dos valores de uma função. A mutação é feita de maneira independente em cada vetor e em cada elemento do cromossomo. O operador de mutação é feito com probabilidade Gaussiana (Distribuição normal), sendo a média 0 e o desvio padrão correspondendo ao gene do pai (GABRIEL; DELBEM, 2008).

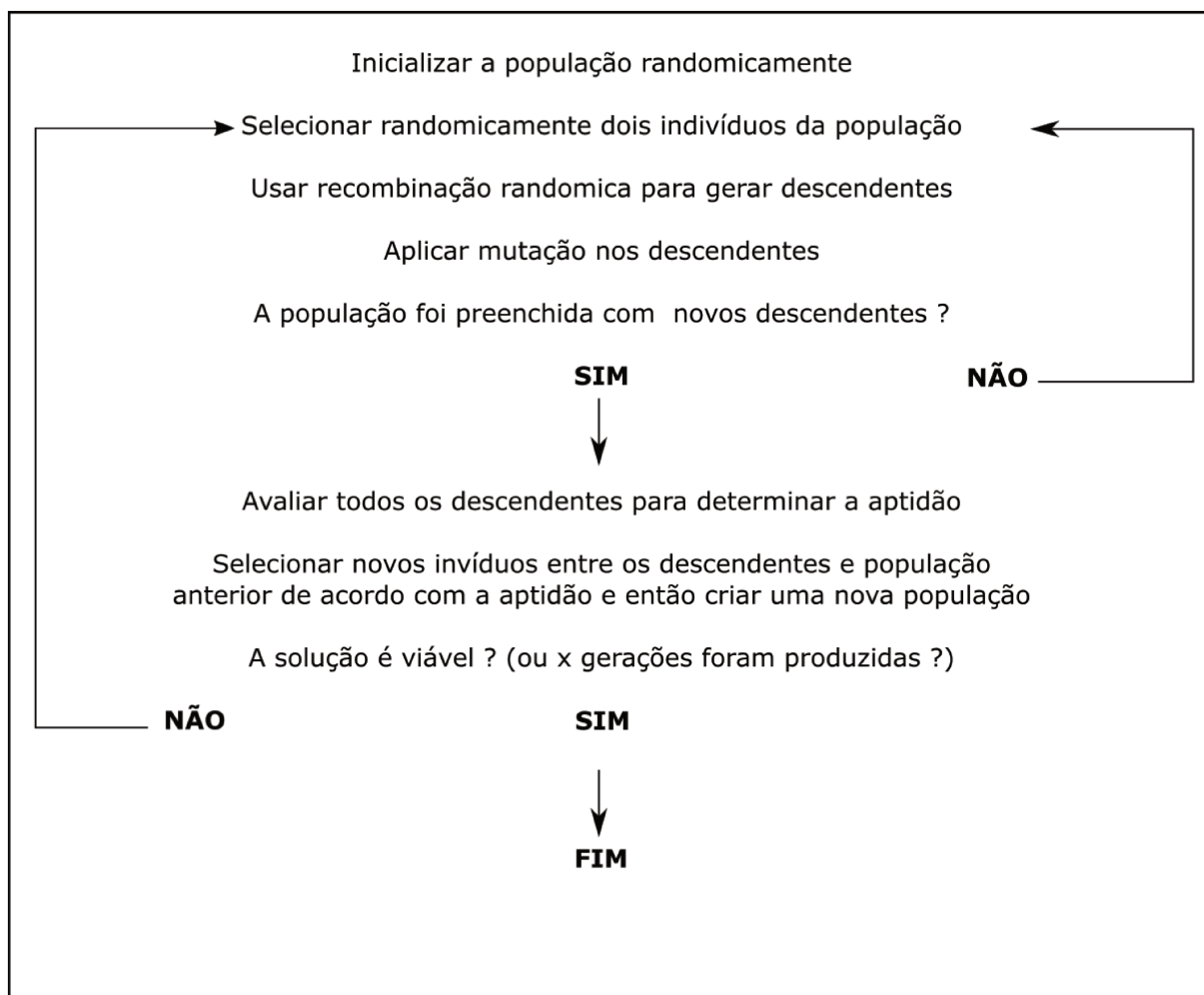
O filho é avaliado, e caso seja mais apto que a solução pai, a solução filho sobrevive e transforma-se na solução pai, caso contrário o filho é descartado e o a solução pai original passa novamente pela operação de mutação para gerar um novo filho (BENTLEY, 1999).

O algoritmo original de estratégia evolutiva tem convergência lenta e está suscetível à estagnação em um ótimo local (JONG, 2006; BENTLEY, 1999) por isso foram desenvolvidos outros métodos de ES na década de 80 (BÄCK; SCHWEFEL, 1996) denominados $(\mu, \lambda) - EE$ e $(\mu + \lambda) - EE$.

O primeiro método, $(\mu, \lambda) - EE$, é definido por : λ que são os indivíduos atuais e μ que é a população gerada (GABRIEL; DELBEM, 2008).

O modelo $(\mu, \lambda) - EE$ seleciona somente os melhores μ indivíduos apenas da solução filho, enquanto o modelo $(\mu + \lambda) - EE$ seleciona os melhores μ indivíduos tanto da solução filho quanto da solução pai (λ) (BENTLEY, 1999).

Figura 31 – Fluxograma do funcionamento das estratégias evolutivas



Fonte: Adaptado de “*Evolutionary Design by Computers* “(BENTLEY, 1999)

3.2.3 Programação Evolutiva

A Programação Evolutiva foi (PE) proposta inicialmente por Fogel (1962) e tinha como objetivo utilizar conceitos da evolução para o desenvolvimento da Inteligência Artificial (IA) (GABRIEL; DELBEM, 2008).

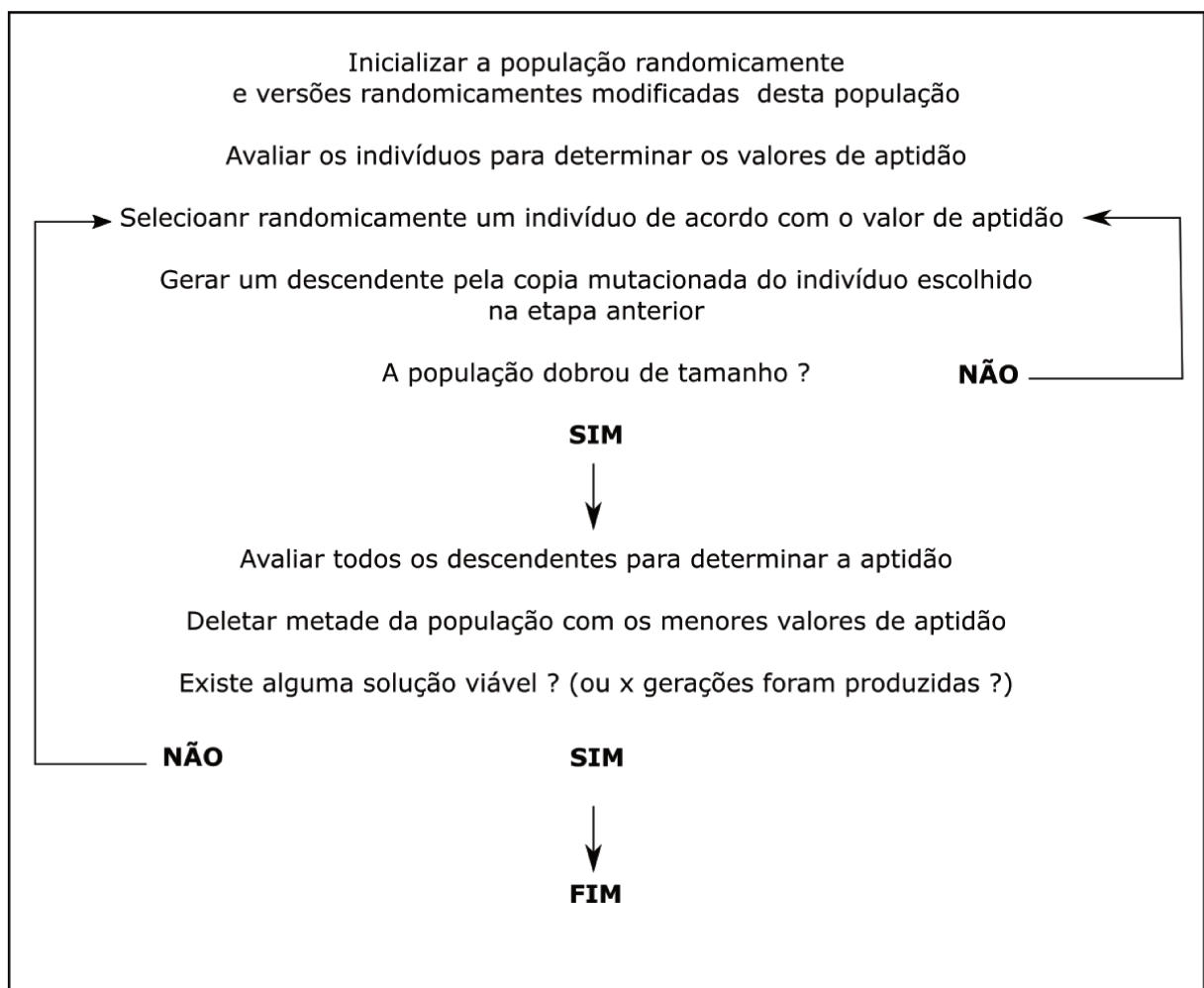
A representação de cada indivíduo em uma PE é feita através de uma máquina de estados finitos (MEF), que processa uma sequência de símbolos (GABRIEL; DELBEM, 2008). A aptidão dos indivíduos está baseada em quanto a sequência de letras de saída gerada por indivíduo se assemelha com a sequência desejada, definida por uma função objetivo. Apenas uma população é mantida e o único operador de diversidade é a mutação, sendo que todos os indivíduos da população geram novos descendentes (BENTLEY, 1999; GABRIEL; DELBEM, 2008). O método de PE garante que todos os indivíduos produzam descendentes, sendo que somente os melhores indivíduos dentre pais e filhos sobrevivem.

Um dos grandes avanços nos estudos de PE foi a introdução do conceito de auto

adaptação, que foi essencial para a classificação das PE. Atualmente os métodos são divididos em 3 tipos: PE padrão, meta-PE e Rmeta-PE (BENTLEY, 1999).

A PE padrão não utiliza nenhum tipo de auto adaptação. O método de meta-PE incorpora mudança de valores dos parâmetros de variação nos indivíduos, permitindo a auto adaptação, enquanto o Rmeta-PE incorpora, no processo de mutação, variação i e parâmetros de covariação j (j é interdependente de i) nos indivíduos, sendo que esse processo permite uma auto adaptação mais precisa. A distinção entre esses métodos está nos níveis de permissão do uso da auto adaptação (BENTLEY, 1999).

Figura 32 – Fluxograma do funcionamento da programação evolutiva



Fonte: Adaptado de “*Evolutionary Design by Computers* “(BENTLEY, 1999)

3.2.4 Programação Genética

A programação genética (PG) pode ser entendida como uma extensão dos algoritmos genéticos, através de métodos de representação dos genomas como árvores, grafos ou conjunto de caracteres *strings*. Utiliza altas taxas de recombinação, e por vezes altas taxas

de mutação. Outra característica importante é a possibilidade de não manter um tamanho fixo de cromossomo em mesma população (GABRIEL; DELBEM, 2008; BENTLEY, 1999; BANZHAF *et al.*, 1998)

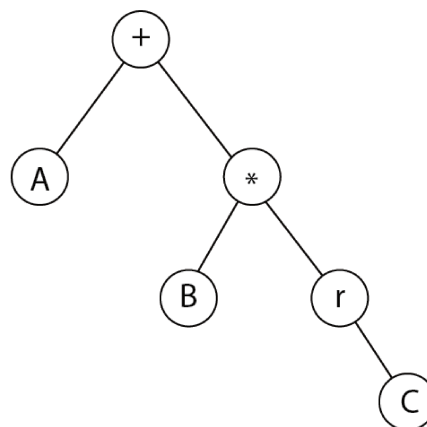
A PG foi desenvolvida por Koza (1992), com a intenção de fazer com que computadores programassem a si mesmos, através do processo de evoluir programas (BENTLEY, 1999)

Existem características que são compartilhadas pela maioria dos PG (BANZHAF *et al.*, 1998):

1. PG é um **processo estocástico** que usa números randômicos para simular o processo randômico da evolução natural.
2. Os programas de PG são construídos através de uma **estrutura** genômica, com tamanho variável, a partir de unidades básicas denominadas funções e procedimentos. Funções aplicam operações nos dados de entrada, estes dados podem ser procedimentos ou dados de saída de outras funções.
3. Os **operadores genéticos** transformam os programas iniciais na população. A recombinação entre dois programas é o principal operador genético, o outro operador é a mutação, como nos Algoritmos Genéticos.
4. A escolha dos indivíduos da população que passam para a próxima geração é feita através de uma função objetivo.

Os programas em PG seguem basicamente os mesmos passos de um AG, mas uma das principais características de um algoritmo de PG é que as soluções são representadas de modo hierárquico.

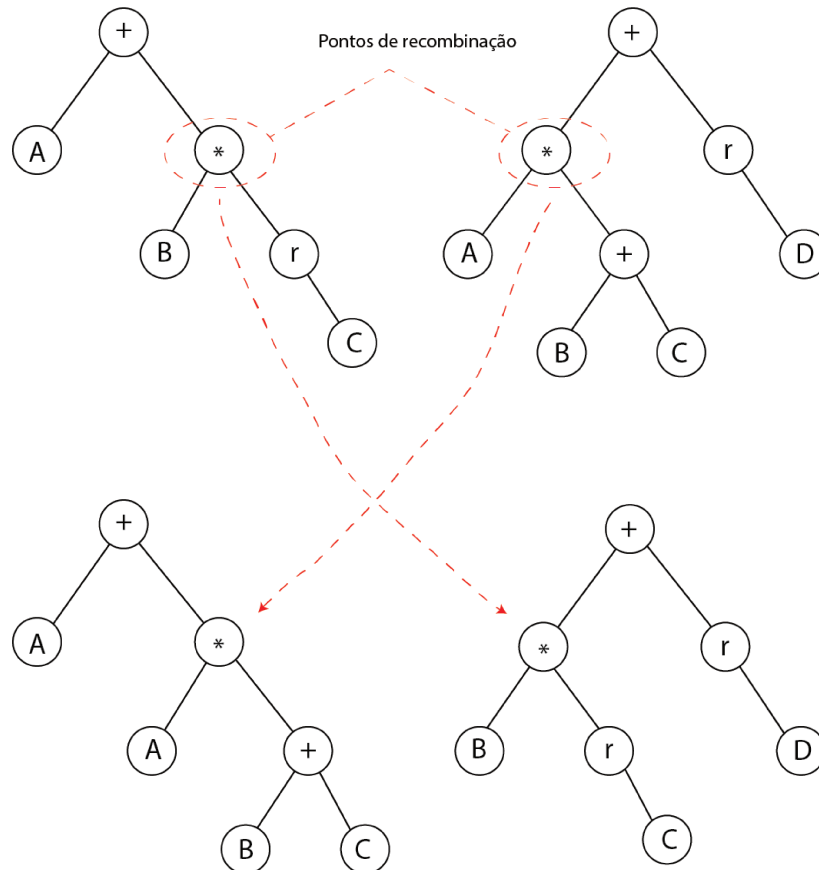
Figura 33 – Representação hierárquica de um cromossomo



Fonte: Elaborado pelo autor

Um dos motivos de se utilizar a representação hierárquica é que através dela é possível aplicar a recombinação em cromossomos que diferem em tamanho, o que não é possível quando se trata de um cromossomo fixo como os dos Algoritmos Genéticos (BENTLEY, 1999).

Figura 34 – Recombinação PG



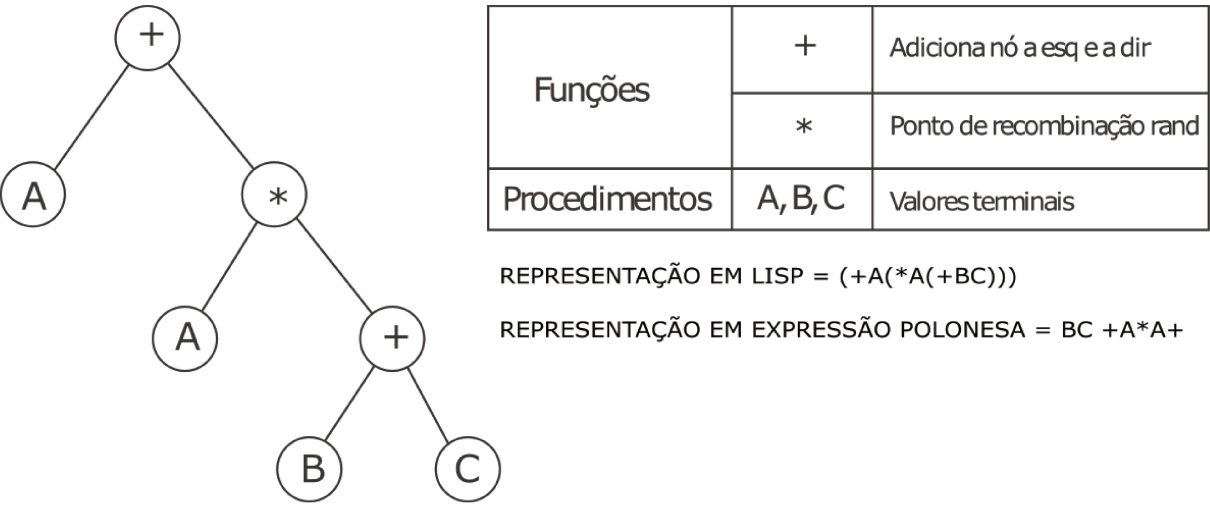
Fonte: Elaborado pelo autor

3.2.4.1 Funções e Procedimentos

As **funções** e **procedimentos** são os elementos básicos sobre os quais o algoritmo de PG é desenvolvido. Os procedimentos providenciam os valores do sistema, enquanto as funções processam esses valores. Quando estão juntos são chamados de **nós** (BANZHAF *et al.*, 1998).

As funções podem ser expressas através de *S-expression*, que é uma linguagem simbólica, na qual símbolos representam determinadas operações das funções. Sendo assim, o cromossomo pode ser implementado através de uma linguagem LISP ou mesmo expresso através de uma expressão polonesa, que é uma forma de codificação em que não é necessária a utilização de parênteses para estabelecer a ordem de execução, o que a torna muito útil para programas de computador.

Figura 35 – Funções e Procedimentos de uma PG através de linguagem LISP e expressão polonesa



Fonte: Elaborado pelo autor

3.3 O projeto arquitetônico com abordagem evolutiva

O projeto de arquitetura baseado em princípios da computação evolutiva tem seu início com as investigações desenvolvidas por Frazer e seus alunos na década de 60. Através de seu livro “*An Evolutionary Architecture*”, publicado em 1975, Frazer discorre sobre uma arquitetura evolutiva fundamentada em processos de geração da forma mediante estratégias evolutivas (FRAZER, 1975). Para Frazer, seria possível simular algumas características e forças da natureza através de um modelo digital, e através dele gerar possibilidades formais para a arquitetura. Assim como, no mesmo período, Nicholas Negroponte estava formulando a idéia uma máquina evolutiva capaz de gerar arquitetura, uma das primeiras hipóteses do uso de inteligência artificial em arquitetura (NEGROPONTE, 1970).

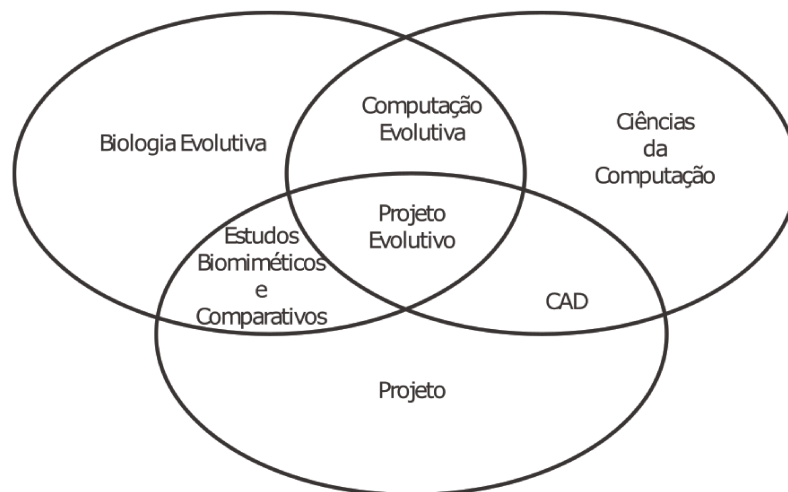
Sobre o cenário brasileiro quanto às pesquisas desenvolvidas sobre essa temática, as teses desenvolvidas ainda são escassas. Jarryer Martino, que fez parte do grupo de pesquisas do LAPAC (laboratório de automação para arquitetura e construção) da UNICAMP, foi um dos pesquisadores pioneiros a estudar o tema. Para Martino (2015), a aproximação da computação evolutiva com métodos de projeto se deu além das coincidências temporais (década de 50 e 60) entre as discussões sobre métodos de projeto (*Design Methods*) e as bases teóricas da computação evolutiva. O entendimento do processo de projeto como um problema pernicioso (*will-defined*) de multi-objetivos estava próximo das características dos objetivos de que os algoritmos evolutivos estavam destinados a resolver, o que colaborou para a aproximação desses campos.

3.3.1 Evolutionary Design by Computers – Bentley e os algoritmos evolutivos

Bentley foi um dos pioneiros a tratar da temática de algoritmos evolutivos voltados à produção de projetos de arquitetura. Em seu livro “*Evolutionary design by computers*” (BENTLEY, 1999), introduz um uma variedade de conceitos e possibilidades de algoritmos evolutivos para a produção de projetos destinados a distintos objetivos que vão desde a otimização de elementos até a produção criativa de objetos arte.

De acordo com Bentley (1999) um projeto evolutivo se enquadra entre as ciências da computação, design e biologia evolutiva (figura 36). O uso de algoritmos evolutivos para a geração de projetos pode ser entendido dentro de 4 categorias principais : otimização do projeto evolutivo *evolutionary design optimization*, projeto evolutivo criativo *creative evolutionary design*, algoritmo evolutivo com abordagem artística *evolutionary art* e algoritmo evolutivo com formas artificiais de vida *evolutionary artificial life forms* (BENTLEY, 1999).

Figura 36 – Projeto evolutivo como consequência da interseção de distintas áreas



Fonte: Adaptado de “*Evolutionary Design by Computers* “(BENTLEY, 1999)

Bentley (1999) entendeu, através da teoria universal darwinista, que existem procedimentos comuns a todos os algoritmos evolutivos: **reprodução**, **herança**, **variação** e **seleção**. Esses algoritmos geram a reprodução dos indivíduos através de operadores de recombinação e mutação, o que permite que as próximas gerações sejam construídas com herança e variação e utilizem-se de algum critério de seleção para determinar quais indivíduos da população devem sobreviver e quais não (BENTLEY, 1999).

3.3.1.1 Otimização do Projeto Evolutivo (*Evolutionary Design Optimization*)

O projeto evolutivo de otimização é o processo que se dá na existência de um projeto pré-determinado, sendo que a intenção da função objetivo é **otimizar** alguns

parâmetros de projeto através de interações baseadas em um algoritmo evolutivo. De acordo com Bentley (1999) esse processo não deve ser classificado como generativo ou criativo, pois é baseado em um projeto existente, em que as partes foram parametrizadas para serem otimizadas através de um algoritmo evolutivo (BENTLEY, 1999).

3.3.1.2 Projeto Evolutivo Criativo (*creative evolutionary design*)

A abordagem evolutiva criativa explora o algoritmo evolutivo como recurso criativo. Esse método está associado às fases iniciais do processo de projeto (MARTINO, 2015), e está dedicado a buscar no espaço de soluções a maior quantidade de soluções através do aumento da variedade das possíveis combinações entre valores e variáveis.

Bentley (1999) dividiu o projeto evolutivo criativo em duas categorias: o **projeto evolutivo conceitual** e o **projeto evolutivo generativo**, sendo o nível de representação o que os diferencia.

O **projeto evolutivo conceitual** define uma estrutura geral para o sistema, estabelecendo como sistema e componentes devem interagir para atingir um resultado desejado.

Na abordagem evolutiva generativa a ênfase da função objetivo é gerar novidade e originalidade, o que permite explorar as diferentes combinações estruturais dos elementos que compõem o sistema (MARTINO, 2015)

3.3.1.3 Algoritmo evolutivo com abordagem artística e formas artificiais de vida (*Evolutionary art / Evolutionary artificial life forms*)

O algoritmo evolutivo com abordagem artística tem como objetivo explorar a geração de peças de arte com um alto apelo estético. As populações nesse tipo de abordagem são geralmente pequenas para permitir que sejam avaliadas rapidamente, e muitas vezes o processo é feito de modo interativo para que o usuário possa avaliar (usuário interagindo com a função objetivo), determinando os membros mais aptos da população (BENTLEY, 1999).

O algoritmo evolutivo com formas artificiais de vida se baseia no desenvolvimento das formas de vida da natureza para criar seres digitais (BENTLEY, 1999). O algoritmo evolutivo nesse caso está dirigido à simulação do comportamento dos seres vivos, sendo que irá simular uma série de tomada de decisões gerando comportamentos que são consequência de um estímulo externo (MARTINO, 2015).

3.4 O algoritmo evolutivo e algumas aplicações

Em mais de 50 anos de projeto evolutivo vimos, através de Bentley (1999), que existem distintas abordagens para se desenvolver um projeto. Nesta seção iremos analisar dois trabalhos com características e objetivos distintos e que foram desenvolvidos através de uma abordagem evolutiva.

3.4.1 Exploração criativa para o desenho de uma cobertura - Jarryer Martino 2015

Esse trabalho foi desenvolvido como um exercício de projeto da tese de doutoramento de Martino (2015), e corresponde à criação de uma cobertura para uma edificação já existente.

O projeto da cobertura é composto por aberturas ao longo de uma superfície com o intuito de explorar a ventilação natural (MARTINO, 2015). O espaço em que o experimento da cobertura foi realizado recebe insolação no período entre 14 e 16 horas.

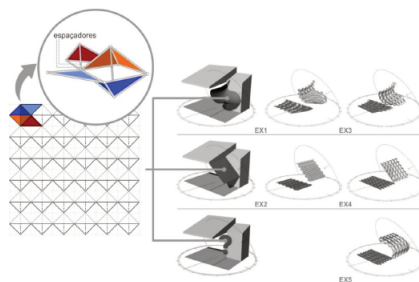
A função objetivo foi desenvolvida para maximizar a área de sombreamento e maximizar o vão máximo para aberturas na superfície. A intenção de Martino buscou resolver uma problemática perniciosa, pois a otimização está buscando otimizar dois objetivos, sombreamento e maximização de abertura, que apresentam funções objetivos que vão em direção opostas, o seja, a função objetivo deverá buscar um valor de equilíbrio entre as partes.

A implementação foi feita através do *plug-in Grasshopper* e utilizando recursos de um outro *plug-in* denominado Geco, sendo que esse último faz interação com o *software Ecotec* da *Autodesk*, desenvolvido para simulações e análises de desempenho ambiental de edifícios.

O desenvolvimento do exercício foi feito através de cinco experimentos. Foram adotadas duas abordagens para a concepção da cobertura. Na primeira abordagem, a cobertura teve alguns de seus fixos e outros variáveis, enquanto no caso da segunda abordagem todos os elementos eram variáveis (MARTINO, 2015). A primeira abordagem foi utilizada nos 4 primeiros experimentos.

Os cinco experimentos adotaram uma, dentre três tipos de superfícies: linear, curvilínea e parametrizada, que definiram a forma da cobertura. A superfície foi transformada em uma malha contendo 12 por 6 módulos, esses módulos serviram de base para a estrutura tridimensional triangulada utilizando “os vértices e as mediatrizes das laterais dos módulos como pontos de conexão dos elementos estruturais.” (MARTINO, 2015).

Figura 37 – Espaçadores, cobertura e experimentos de Jarryer Martino

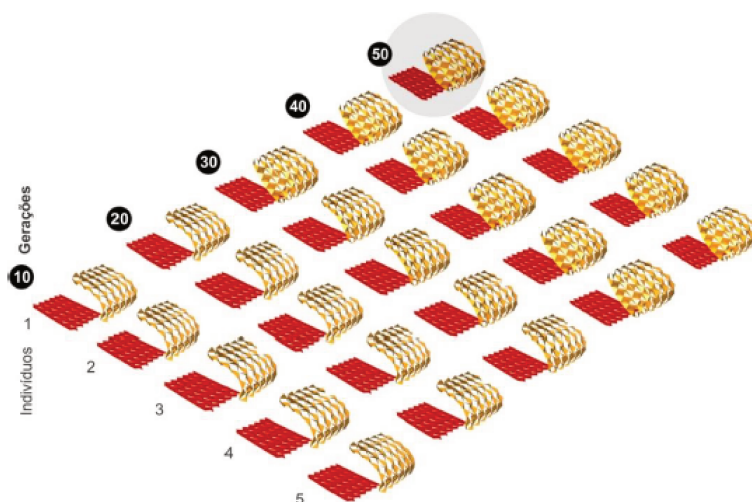


Fonte: “Algoritmos evolutivos como método para desenvolvimento de projetos de arquitetura” (MARTINO, 2015)

O dimensionamento da abertura é definido por espaçadores que variam de tamanho em função da distância entre sua posição na malha e o ponto de atração mais próximo.

Foram geradas 2250 possibilidades de cobertura, através de 50 gerações com 50 indivíduos somados aos 50 indivíduos da geração inicial.

Figura 38 – População de cobertura através de gerações



Fonte: “Algoritmos evolutivos como método para desenvolvimento de projetos de arquitetura” (MARTINO, 2015)

Os experimentos 1 e 2 tinham as aberturas voltadas para oeste e tiveram soluções com menor área de sombreamento. Os gráficos de área de sombreamento e dimensão demonstrou uma relação inversa na somatória da função objetivo, ou seja, em soluções de pouco sombreamento havia mais aberturas e vice-versa. Isso demonstrou que o algoritmo evolutivo encontrou dificuldades para equilibrar a função objetivo.

Figura 39 – Gráfico das áreas de sombreamento em m^2 vs gráfico do somatório dos vãos em metros.

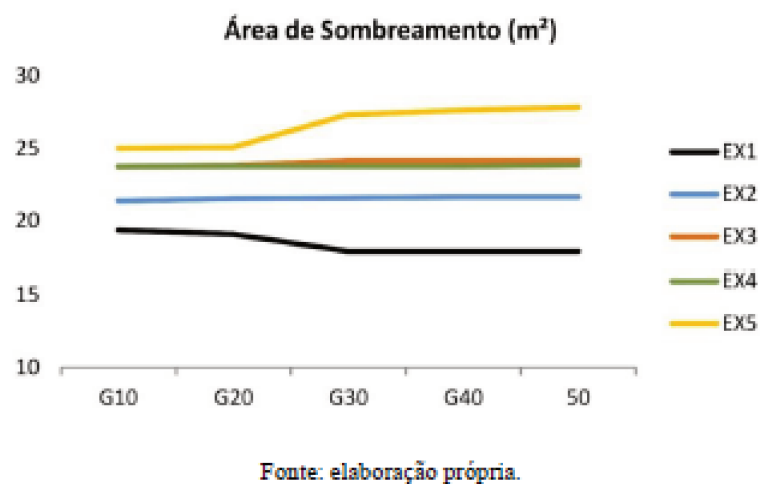
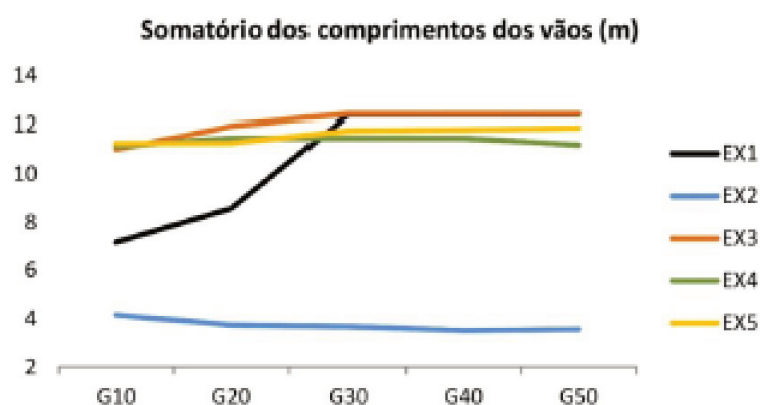
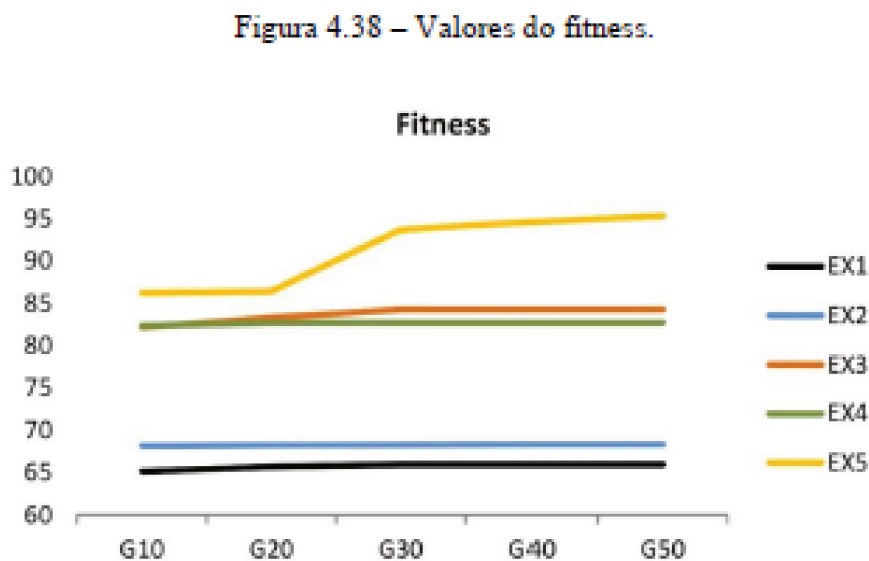


Figura 4.37 – Somatório do comprimento dos vãos em metros.



Fonte: “Algoritmos evolutivos como método para desenvolvimento de projetos de arquitetura” (MARTINO, 2015)

A mudança de orientação das aberturas para norte colaborou para os experimentos 3 e 4, mas foi no experimento 5 em que foi utilizado uma abordagem totalmente paramétrica para a cobertura que a função objetivo atingiu seu melhor desempenho (MARTINO, 2015).

Figura 40 – Valores de *fitness* dos experimentos

Fonte: “Algoritmos evolutivos como método para desenvolvimento de projetos de arquitetura” (MARTINO, 2015)

O experimento 5 demonstrou uma grande evolução se comparado aos outros. Apresentou um aumento de 9,6% de desempenho, em soma ponderada de ventilação e sombreamento, enquanto os outros variavam entre 0,3 % e 2,5%, sendo que todos os objetivos tinham pesos iguais (figura 39).

Como conclusão, Martino discorre sobre a possibilidade de compreensão do processo evolutivo de projeto como aliado no entendimento das variáveis arquitetônicas e visualização dos dados do problema. Para Martino, o projeto é visto de um modo diferenciado, em que “a informação passa a ser a matéria prima a ser gerenciada, manipulada, avaliada e tratada, apresentando um peso considerável durante todo o processo de projeto” (MARTINO, 2015).

3.4.2 Generative Programming and Spatial Morphogenesis- Paul Coates e Dimitrios Makris -(1999)

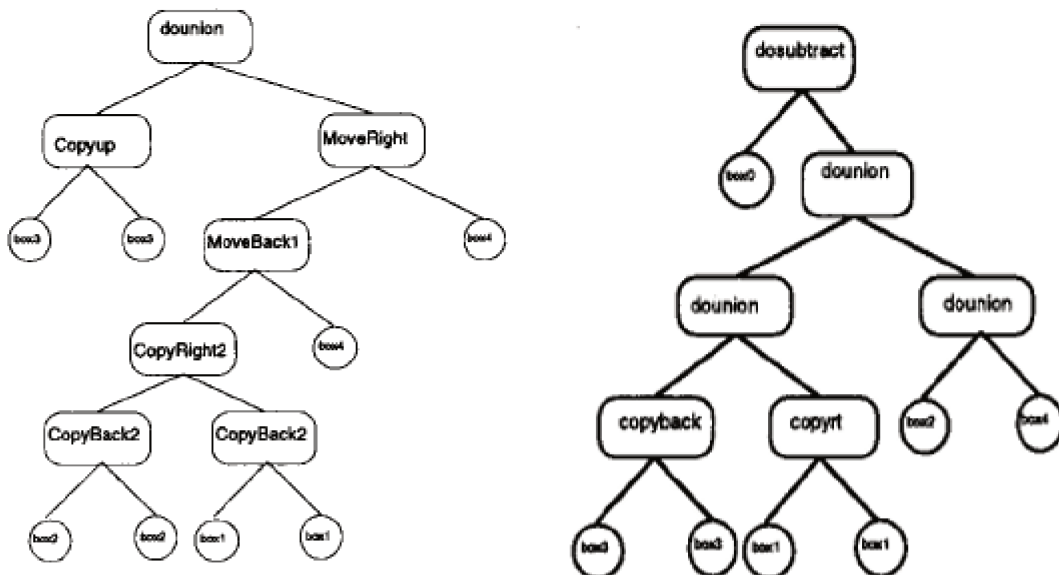
Neste trabalho, Coates e Makris exploram as possibilidades da programação genética através do uso de uma gramática de forma, através da desconstrução dos elementos que constituem o sistema Dom-ino de Le Corbusier e evoluem possibilidades de reinterpretações do sistema. O primeiro passo do trabalho foi entender o sistema Dom-ino como nós de um programa de PG, que é composto por procedimentos e funções, e a partir desse entendimento construir um modelo hierarquizado do desenho original.

Tabela 4 – Funções utilizadas no método

copyUP1i	moveRT1	
copyUP2	moveBack1	
copyDN1	moveBack2	
copyRT1	moveUP2	
copyback1	moveDN1	
copyback2	moveRT2	
copyRT2		
dounion	dosub	doint

Fonte: “*Genetic Programming and Spatial Morphogenesis*” (COATES; MAKRIS, 1999)

Figura 41 – Área de sombreamento Vs comprimento dos vãos



“*Genetic Programming and Spatial Morphogenesis*”(COATES; MAKRIS, 1999)

Foram feitos 4 experimentos nesse trabalho, que serão descritos a seguir.

Experimento 1

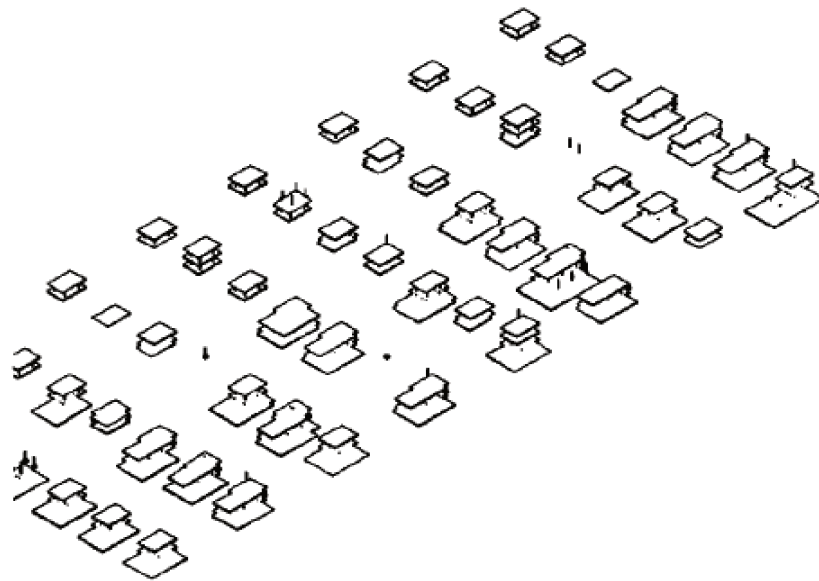
O primeiro experimento foi realizado por Coates e Makris por translação e cópia dos elementos (*Block move & copy*), que foi chamado de modelo aditivo. As funções eram expressões como copiar em cima, mover para baixo, etc (COATES; MAKRIS, 1999) e correspondiam a operações da gramática. O sistema Dom-ino foi definido através de 3 objetos, mais um *box extra*.

Os nós que correspondiam a funções eram 13, dos quais 10 eram a expressão original, e 3 que eram operações booleanas.

Os operadores terminais eram quatro cuboides: **Box1**, **Box2**, **Box3** e **Box4**.

A população inicial foi selecionada manualmente para a expressão Dom-ino. Na primeira geração todos os indivíduos eram geneticamente idênticos, e as próximas gerações eram escolhidas pelo usuário de acordo com seu julgamento subjetivo de quais eram os mais aptos a passar para a próxima geração.

Figura 42 – População gerada através do método aditivo



Fonte: “*Genetic Programming and Spatial Morphogenesis*”(COATES; MAKRIS, 1999)

Após 7 gerações (figura 43) do usuário, as populações foram lentamente aumentando a variedade produzida . Usando procedimentos de recombinação e mutação foi possível explorar uma diversidade formal que iam de um extremo a outro (COATES; MAKRIS, 1999).

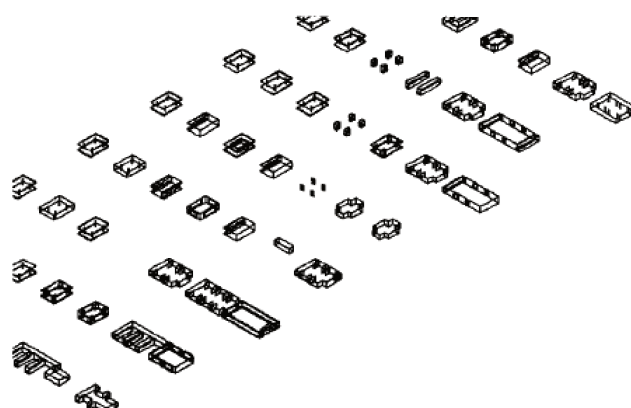
Experimento 2

O segundo experimento foi feito através de operações booleanas, em que se utilizava um grande bloco do qual eram subtraídos partes até que se aproximasse da morfologia do sistema Dom-ino. O sistema era constituído por 5 blocos, o bloco zero de onde seriam subtraídos partes e também os blocos de 1 a 4, que seriam copiados, unidos para subtrair o bloco 0.

Os nós de GP que correspondiam a funções desse experimento eram os mesmos do experimento 1, sendo que os operadores terminais eram 3 cuboides: **Box0**, **Box1** e **Box2**.

Nesse método a criação da estrutura Dom-ino é baseada em uma série muito pequena de funções (COATES; MAKRIS, 1999), em que os blocos de subtração (1-4) se movem para além do bloco positivo 0, e assim que sobrevivem formam aglomerações de grandes volumes.

Figura 43 – População gerada através do método subtrativo

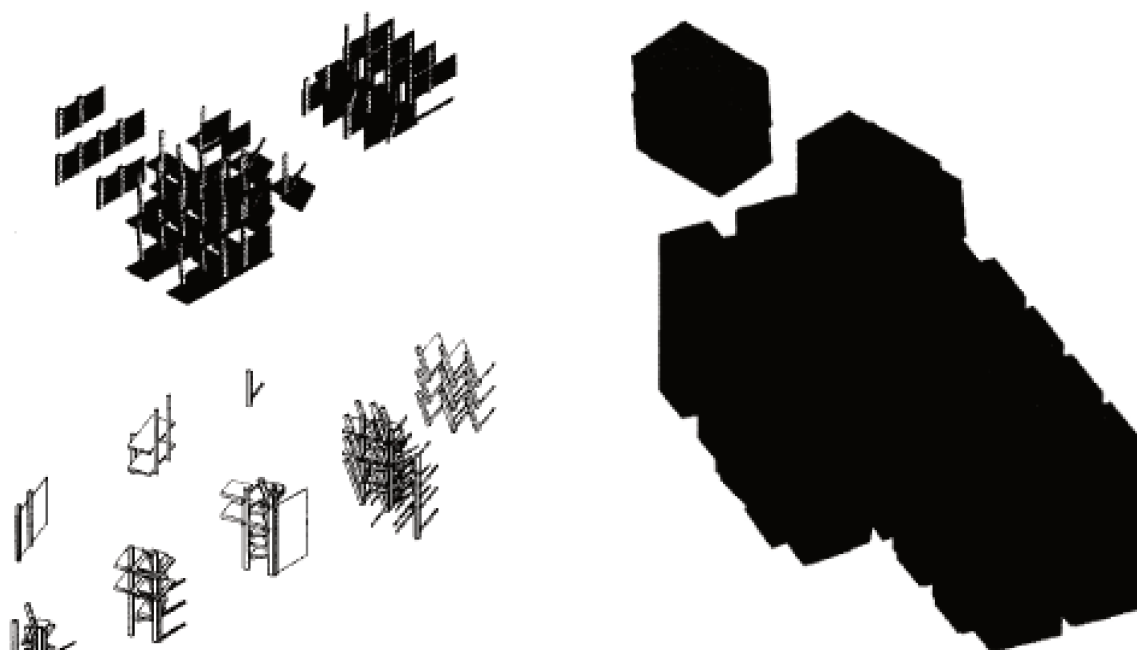


Fonte: “*Genetic Programming and Spatial Morphogenesis*”(COATES; MAKRIS, 1999)

Coates e Makris identificaram que o segundo experimento gerava formas que poderiam ser úteis para composições espaciais. Através de mais dois experimentos modificaram a gramática para uma que explorasse novamente os conceitos aditivos, mas que dessa vez estava ligada à ideia de gerar espaços que refletissem certas qualidades da arquitetura doméstica japonesa. O outro experimento era uma gramática que explorava os conceitos de cheios e vazios, e desenvolvia a ideia de sobreposição entre similaridades com sutis deslocamentos entre os cuboides (COATES; MAKRIS, 1999).

Foram testadas algumas funções objetivos que avaliaram a qualidade dos resultados dos invólucros produzidos pelas soluções. No modelo sólido, a função objetivo dependia da configuração global do modelo, através de cálculos de índices de tamanho e forma.

Figura 44 – Resultados obtidos através dos experimentos aditivos novos e método de cheios e vazios



Fonte: “*Genetic Programming and Spatial Morphogenesis*”(COATES; MAKRIS, 1999)

Como conclusões, eles observaram a necessidade de mais trabalhos melhorando a eficácia da função objetivo, e uma interface em que o usuário interagisse com o programa seria um próximo passo.

3.5 Considerações sobre os algoritmos evolutivos

Este capítulo tratou dos algoritmos evolutivos pelo viés computacional, apresentando seus métodos e procedimentos. Também apresentou de maneira sucinta a aproximação da computação evolutiva com os métodos de projeto e discorreu sobre o delineamento feito por Bentley (1999) para o projeto evolutivo. Por fim, apresentou 3 trabalhos acadêmicos de arquitetura que exploram a abordagem evolutiva como método de projeto. O próximo capítulo irá discutir 32 artigos de *space planning* com abordagem evolutiva, coletados em bases online de pesquisa (*CUMINCAD*, *IEEXplore*, *ACM digital Library*, *Research Gate*, *Science direct* e *CiteSeerX*).

Parte III

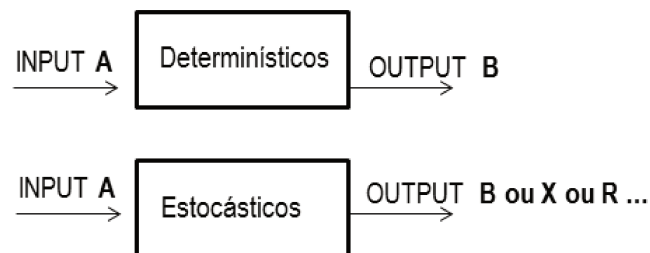
Space Planning com abordagem evolutiva (1992-2014)

4 Space Planning com abordagem evolutiva

No início dos anos 90, muitos artigos foram publicados relacionando estratégias da inteligência artificial (IA) com Arquitetura e Design. Algoritmos evolutivos como os algoritmos genéticos e a programação genética foram alguns dos métodos abordados (TAM, 1992; JO; GERO, 1996; JAGIELSKI; GERO, 1997). Uma das primeiras temáticas que relacionavam IA com design foram as destinadas a solucionar problemas de *space planning*, pois os algoritmos de IA foram fundamentais para o desenvolvimento de soluções para os problemas de combinatória.

De acordo com Gero, os problemas de *space planning* são considerados, por sua complexidade computacional, um NP-completo (GERO; KAZAKOV, 1997b) ou seja, podem ser solucionados por uma máquina não determinística (estocástica) (figura 45) em tempo polinomial. Entretanto, devido às dificuldades associadas a esse tipo de classe de problema, é impossível encontrar uma solução ótima através de um algoritmo em um tempo viável, pois o tempo computacional investido é demasiadamente elevado (GERO; KAZAKOV, 1997b). Métodos de busca meta-heurísticos como algoritmos genéticos têm demonstrado sucesso em resolver problemas de otimização combinatória, como os problemas de SP.

Figura 45 – Fluxo de dados de um processo determinístico e de um processo estocástico



Fonte: Elaborado pelo autor

A resolução de problemas de *space planning* através de uma **abordagem evolutiva** é um campo de pesquisa que tem sido estudado desde o início da década de 90, mas devido à amplitude do tema ainda há muito a ser explorado em termos de aplicações na arquitetura, e as novas ferramentas de programação visual podem colaborar na absorção do tema pela nova geração de arquitetos.

Tam (1992) argumenta que algoritmos evolutivos, mais especificamente os algoritmos genéticos (AG), são diferentes da maioria dos métodos de busca que utilizam o espaço de soluções através de uma busca pontual. O mecanismo de um AG é desenvolvido

para explorar uma população de pontos, identificando as boas soluções, ou seja, durante o processo de busca a informação útil de toda a população não é descartada, dado que o algoritmo pode aprender no decorrer das gerações, preservando as soluções mais aptas do espaço solução.

4.1 Revisão da Literatura 1992-2014

Nos últimos 15 anos, foram identificadas 3 revisões de literatura relacionando problemas de SP com algoritmos evolutivos. A primeira revisão (LIGGETT, 2000) não é especificamente sobre métodos evolutivos, mas estabelece uma classificação dos problemas de SP mencionando pela primeira vez a abordagem evolutiva. Homayouni (2007) apresenta alguns dos principais trabalhos e métodos para resolução de SP com abordagem evolutiva. Por fim a terceira revisão de literatura, feita por Dutta e Sarthak (2011) foi a revisão mais completa, categorizando e comparando 16 artigos relacionados ao tema.

Entretanto, nenhuma dessas revisões usou bancos de dados *online* para cobrir as publicações desde que se iniciaram as pesquisas nesse campo, sendo assim, um estudo através de uma base de dados dinâmica e atualizada se faz necessário. Nesta seção iremos apresentar uma revisão da literatura de 31 artigos, entre aplicações e revisões de problemas de SP com abordagem evolutiva. As bases de dados utilizadas para a seleção de artigos foram : *CUMINCAD*, *IEEXplore*, *ACM digital Library*, *Research Gate*, *Science direct* e *CiteSeerX*.

O método utilizado para a seleção dos artigos seguiu os seguintes passos:

1. Identificação nas bases de dados online (*CUMINCAD*, *IEEXplore*, *ACM digital Library*, *Research Gate*, *Science direct* e *CiteSeerX*) utilizando palavras-chave específicas: “*facility layout problem*”, “*space planning*” e “*space layout planning*”.
2. Combinação das expressões: “*genetic algorithm*”, “*evolutionary*” e “*genetic programming*” com as palavras-chaves do passo anterior para refinar a busca.
3. Análise da bibliografia dos artigos resultantes para se identificar novos artigos em potencial.
4. Foram descartadas algumas publicações, por serem consideradas fora do escopo arquitetônico.
5. Revisão crítica dos artigos selecionados.
6. Classificação e comparação de cada publicação por: método evolutivo, método de space planning, o tipo de representação algorítmica usada pelo método evolutivo e o tipo de problema de space planning resolvido.

Por definição do recorte proposto, foram selecionados somente artigos em língua inglesa, devido à ampla divulgação e reconhecimento internacional como padrão de divulgação científica. A tabela 5 apresenta os artigos selecionados para análise em ordem cronológica:

Tabela 5 – Relação de artigos analisados

AUTORES	ANO	ARTIGOS	BASE DE DADOS	PUBLICAÇÃO
TAM	1992	Genetic algorithms, function optimization	SCIENCE DIRECT	Journal of Operation Research
KADO	1995	An investigation of genetic algorithms for facility layout problems	CITeseerX	Thesis
JO e GERO	1995	Representation and use of design knowledge in Evolutionary Design	CUMINCAD	CAAD Futures
GARCEZ-PEREZ, SCHONEFELD e WAINWRIGHT	1996	Solving facility layout problems using genetic programming	CITeseerX	Proceedings of the 1st annual conference on genetic programming
MELLER e GAU	1996	The facility layout problem: Recent and emerging trends and perspectives	SCIENCE DIRECT	Journal of Manufacturing Systems
GERO e KAZAKOV	1997	Learning and re-using information inspace layout planning problems using genetic engineering	SCIENCE DIRECT	Artificial Intelligence in Engineering
DAMSKI e GERO	1997	An evolutionary approach to generating constraint-based space layout topologies	CUMINCAD	CAAD Futures
JAGIELSKI e GERO	1997	A genetic programming approach to the space layout planning problem	CUMINCAD	CAAD Futures
GERO e KAZAKOV	1997	Evolving design genes in space layout planning problems	CITeseerX	Artificial Intelligence in Engineering
JO e GERO	1998	Space layout planning using an evolutionary approach	SCIENCE DIRECT	Artificial Intelligence in Engineering

LI e LOVE	1998	Site-level facilities layout using genetic algorithms	RESEARCHGATE	Journal of Computing in Civil Engineering
KOCHHAR, FOSTER e HERAGU	1998	Hope: A genetic algorithm for the unequal area facility layout problem	SCIENCE DIRECT	Computers and Operations Research
TAVAKKOLI, MOGHADDAIN e SHAYAN	1998	Facilities layout design by genetic algorithms	SCIENCE DIRECT	Computers Industrial Engineering
ROSENMAN e GERO	1999	Evolving designs by generating useful complex gene structures	CITeseerX	Evolutionary Design by Computers
GARZA e MAHER	1999	Evolving design layout cases to satisfy Feng Shui constraints	CITeseerX	ACADIA
ELEZKURTAJ e FRANCK	2000	Geometry and Topology : A User-Interface to Artificial Evolution in Architectural Design	CUMINCAD	eCAADe
LIGGETT	2000	Automated facilities layout: past, present and future	CUMINCAD	Automation in Construction
MICHALEK e PAPALAMBROS	2002	Interactive design optimization of architectural layouts	CITeseerX	Engineering Optimization
HARMANANI, ZOUEN e HAJAR	2004	A parallel genetic algorithm for the geometrically constrained site layout problem with unequal-size facilities	CITeseerX	International Journal of Computational Intelligence and Applications
HOMAYOUNI	2007	A genetic algorithm approach to space layout planning optimization	CITeseerX	Thesis
DOULGERAKIS	2007	Genetic programming + unfolding embryology in automated layout planning	RESEARCHGATg	Thesis
BANERJEE e QUIROZ	2008	A model of creative design using collaborative interactive genetic algorithms	CITeseerX	DESIGN COMPUTING AND COGNITION
WONG e CHAN	2009	EvoArch: An evolutionary algorithm for architectural layout design	SCIENCE DIRECT	Computer-Aided Design

KNECHT	2010	Generating floor plan layouts with k-d trees and evolutionary algorithms	RESEARCHGATE	GA2010 – XIII Generative Art Conference
FLACK	2010	Evolution of Architectural Floor Plans	RESEARCHGATE	Thesis
THAKUR e KUMARI	2010	Architectural layout planning using genetic algorithms	IEEEEXPLORER	Computer Science and Information Technology IEEE International Conference
DUTTA and SARTHAK	2011	Architectural space planning using evolutionary computing approaches: a review	ACM DIGITAL LIBRARY	Artificial Intelligence Review
RODRIGUES, GASPAR e GOMES	2013	An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 1: Methodology	SCIENCE DIRECT	Computer-Aided Design
RODRIGUES, GASPAR e GOMES	2013	An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, Part 2: Validation and performance tests	SCIENCE DIRECT	Computer-Aided Design
RODRIGUES, GASPAR e GOMES	2013	An approach to the multi-level space allocation problem in architecture using a hybrid evolutionary technique	SCIENCE DIRECT	Computer-Aided Design
FERNANDO	2014	Space planning and preliminary design using artificial life	CUMINCAD	CAADRIA

Fonte: Elaborado pelo autor

4.2 Slice tree structure (STS) com abordagem evolutiva

O primeiro trabalho a relacionar algoritmos evolutivos e *space planning* (TAM, 1992) identificou o potencial que essa classe de algoritmos teria para a resolução desse tipo de problema. A pesquisa de Tam pontua dois principais problemas encontrados nos procedimentos heurísticos usados até então (TAM, 1992):

- O espaço de busca se desenvolve através de um único ponto no espaço solução.
- Restrições geométricas são difíceis de implementar.

Tanto o método de *Simulating Annealing* quanto outros métodos de busca não aprendem com as interações realizadas no espaço solução. A partir dos problemas encontrados nos demais métodos, Tam (1992) identificou que o método de AG utiliza um procedimento mais efetivo na resolução de problemas de SP, além disso desenvolveu uma linguagem simbólica e hierárquica para a representação de projetos de *layouts* arquitetônicos.

Os *layouts* foram representados através de uma estrutura denominada *Slice Tree Structure* (STS). Esse método foi desenvolvido por Otten (1982), e estabelece as repartições do *layout* através de um processo recursivo e *top-down* de fatiamento de um retângulo (figura 46) (TAM, 1992).

Figura 46 – Representação do fatiamento de um retângulo e sua representação através de uma *slicing tree*.

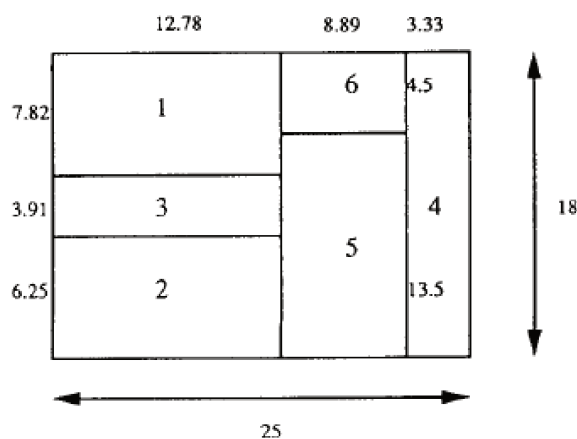


Figure 2. A slicing structure

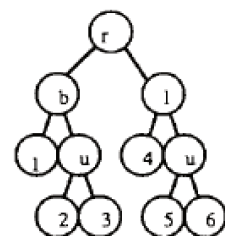


Figure 3. A slicing tree

Fonte: “*Genetic algorithms, function optimization*” (TAM, 1992)

Cada nó da estrutura do sistema *slicing tree* representa uma maneira de repartição dos retângulos através de uma letra. Como a estrutura em árvore do sistema segue uma

hierarquia na representação do *layout*, a troca interna dos nós representa uma sequência de diferentes *layouts*.

Figura 47 – Diferentes operadores representando diferentes maneiras de definição dos *layouts*.

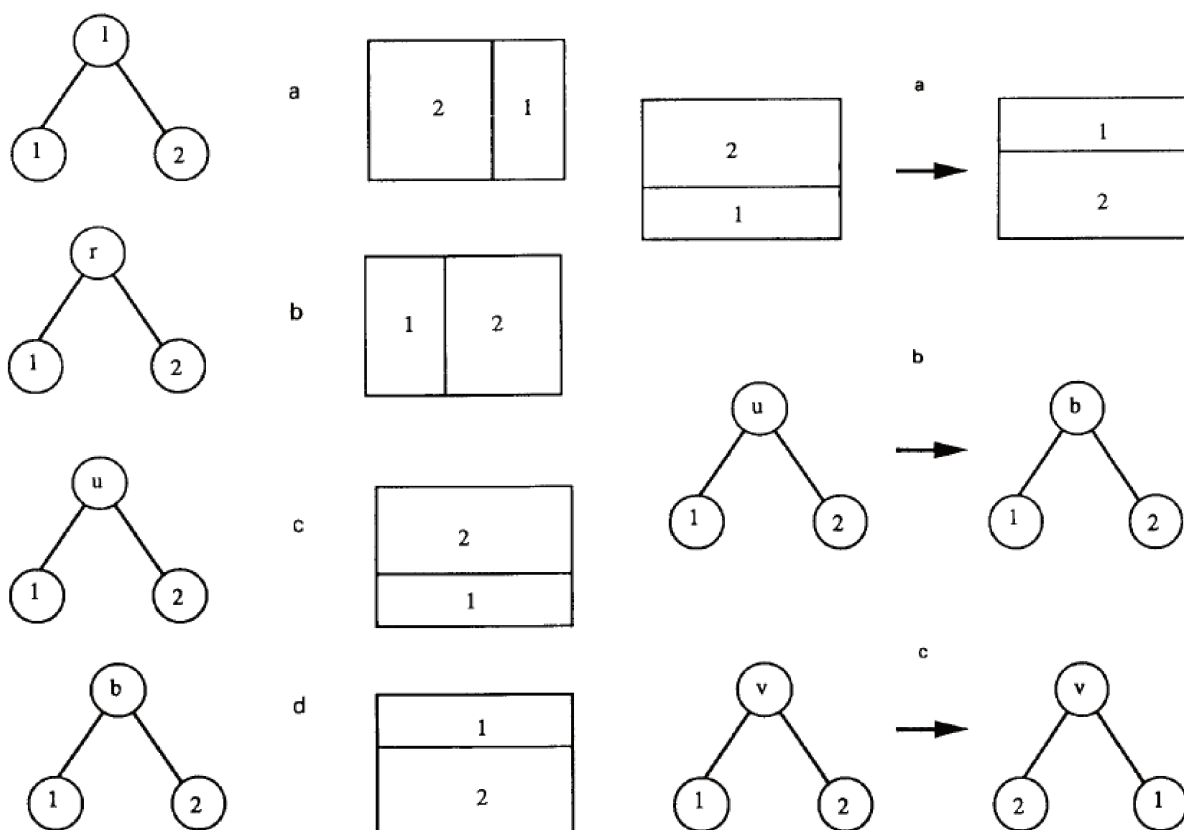


Figure 5.

Figure 6. Cut tree with 2 operators

Fonte: “*Genetic algorithms, function optimization*” (TAM, 1992)

Através do algoritmo de GA, cada indivíduo da população de *layouts* de Tam foi representado através de um cromossomo definido por uma expressão polonesa. A função objetivo estava direcionada a resolver os fluxos de materiais entre os ambientes através do método de *Quadratic assignment Problem* (QAP). Para validar o método, Tam comparou os resultados que foram obtidos em sua pesquisa com outros métodos de busca, como o método de *Simulated Annealing* (SA). Os algoritmos genéticos após 150 gerações, e com 10 dados de entrada diferentes na população inicial, demonstraram melhores resultados tanto nas melhores soluções produzidas quanto na média das soluções produzidas para 12, 15, 20 e 30 ambientes. A figura 48 ilustra uma comparação entre os métodos de *hill climbing* e AG feita por Tam para edifícios de 20 e 30 ambientes.

Tabela 6 – Comparação feita por Kado entre diversos AG para space planning.

Problema	melhores pontuações	média das pontuações	desvio padrão	número de amostras	número de avaliações
Kea91-11	2829.4	2829.4	0	1	-
Kea91-11	2287.041	2287.041	0	1	-
Kea91-16	64	83.5	11.517	10	192200
Kea91-20	153	170.75	13.679	10	219024
TL91-5	246.82	246.82	0	1	tempo de cpu = 0.32 seg
TL91-6	514	514	0	1	tempo de cpu= 0.57 seg
TL91-7	559	559	0	1	tempo de cpu= 4.50 seg
TL91-8	839	839	0	1	tempo de cpu= 12.45 seg
TL91-12	3162	3162	0	1	tempo de cpu= 89.50 seg
TL91-15	5862	5862	0	1	tempo de cpu= 379.63 seg
Tam92-20a	23544	23544	0	1	1593
Tam92-30a		45044	0	1	3712
VCea91-10	24445	24445	0	1	tempo de cpu= 847seg

Fonte: Adaptado de “*An investigation of genetic algorithms for facility layout problems*”

algoritmos evolutivos foi realizado por Kado (1995). Kado faz a comparação dos trabalhos produzidos anteriormente com os métodos de Algoritmos Genéticos, *simulated annealing*, *hill climbing* e *quasi-newton*. Os métodos foram comparados entre diferentes configurações de AG (tabela 6), e com outros métodos de busca heurísticos.

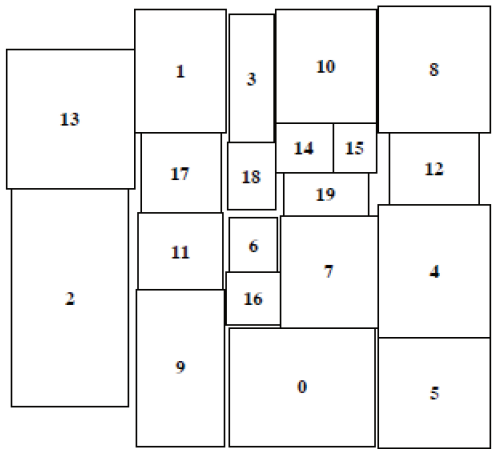
O trabalho de Garces-Perez *et al.* (1996) explorou o sistema de STS com programação genética e fez comparações (figura 49) com os trabalhos anteriores, conseguindo alguns avanços na produção de *layouts* com 20 e 30 ambientes (figura 50). Os *layouts* foram produzidos através de um processo de fatiamento *bottom-up* (figura 51) que já havia sido descrito por Kado (1995).

Figura 49 – Comparações feitas por Perez, Shoenefeld e Wainwright entre métodos de STS.

Problem	TL91	KA95	GPK	GP96	Pop	Gen	First	Cr	Mu	M1	M2	M3
TL91-5	247	228	228	226	600	1,000	24	50	50	15	70	15
TL91-6	514	361	361	384	600	1,000	33	50	50	15	70	15
TL91-7	559	596	615	568	600	1,000	175	50	50	15	70	15
TL91-7	559	596	596	595	1,500	4,000	979	50	50	15	70	15
TL91-8	839	878	878	878	1,500	4,000	1,657	50	50	15	70	15
TL91-12	3,162	3,283	3,576	3,220	1,500	3,000	168	70	40	25	50	25
TL91-15	5,862	7,384	7,612	7,510	1,500	3,000	1,383	70	40	25	50	25
TL91-20	—	16,393	14,026	14,033	1,500	3,000	2,521	50	50	15	70	15
TL91-30	—	41,095	39,176	39,018	1,500	3,000	2,933	75	40	25	50	25

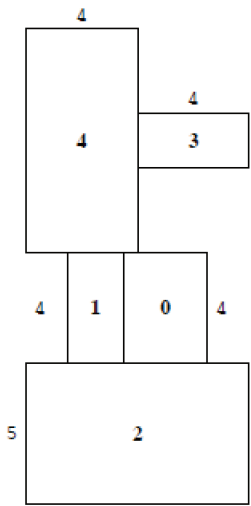
Fonte: “*Solving facility layout problems using genetic programming*” (GARCES-PEREZ *et al.*, 1996)

Figura 50 – Layout produzido por Perez, Shoenefeld e Wainwright para um edifício de 20 ambientes.



Fonte: “Solving facility layout problems using genetic programming” (GARCES-PEREZ *et al.*, 1996)

Figura 51 – Representação do fenótipo *bottom-up* da solução (D(D 2 (R(0 1)(R 3 4))).



Fonte: “Solving facility layout problems using genetic programming” (GARCES-PEREZ *et al.*, 1996)

Em sua dissertação de mestrado Doulgerakis (2007) desenvolveu, através de uma série experimentos, um algoritmo de programação genética que resolvia problemas de *space planning* através do método de STS. O sistema foi implementado em *Processing*, que consiste em uma adaptação da linguagem Java feita por pesquisadores do MIT, muito utilizada por artistas e designers que utilizam programação em suas obras.

Tabela 7 – Comparação entre distintos métodos de seleção realizados por Flack

Objetivo	Geometria	Conexão	Função	Aspecto
Soma ponderada	7.333	0	0.1667	0.0020
Pareto	7.333	13.97	0.03333	1.462
Ranqueamento	7.333	13.97	0.03333	1.462

Fonte: Adaptado de “*Evolution of Architectural Floor Plans*”

A função objetivo de Doulgerakis estava ligada a preferências de localização, conectividade entre ambientes e quantidade de atividade por ambientes. A otimização foi realizada através de uma função multicritério desenvolvida através de dois métodos: otimização por soma ponderada e método ranqueado (figura 52).

Figura 52 – Otimização de soma ponderada e método ranqueado

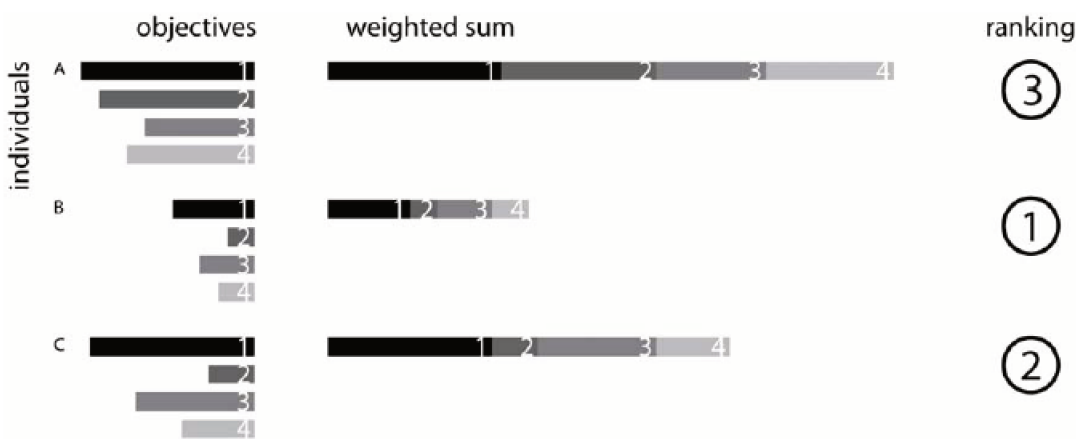


Figure40 Weighted Sum Fitness function

Fonte: “*Genetic programming + unfolding embryology in automated layout planning*” (DOULGERAKIS, 2007)

Flack e Ross (2011) também explorou em sua dissertação de mestrado o uso de programação genética e o sistema de STS, comparando distintos métodos de seleção de indivíduos (tabela 7).

4.3 One-to-One Assignment com abordagem evolutiva

O método de *one-to-one assignment* consiste na alocação de uma lista de ambientes em uma lista de locais (LIGGETT, 2000). Existem dois principais métodos para essa abordagem: o método de **áreas iguais** e o de **áreas desiguais**. No sistema de áreas iguais, um plano é subdividido em partes iguais de forma que cada parte represente um ambiente, enquanto no método de áreas desiguais um plano é subdividido em pequenas unidades, sendo que o ambiente é criado a partir da soma das unidades, de acordo com as áreas requeridas. Jo e Gero (1995) desenvolveram uma gramática de geração de formas em que as regras formais de composição estavam introduzidas no genótipo dos indivíduos. O sistema de composição dos cromossomos foi definido por Jo e Gero como (JO; GERO, 1995):

$g_i = \{\text{local, alelo}\}$

$G_i = \sum g_i$

$P = m(G)$

$p(t) = \{G_1, G_2, G_3 \dots, G_n\}$

Onde: g_i é um gene.

G_i um genótipo em string.

m mapeamento ou um operador de transformação.

P um fenótipo.

$p(t)$ população.

A gramática operava com o sistema *Left-Hand Side* (LHS) – *Right-Hand Side* (RHS), onde LHS representava o elemento do design a ser transformado e RHS o elemento após uma operação de transformação (JO; GERO, 1995):

LHS então RSH ou $LHS \rightarrow RSH$

$\{S_r = E_x, (E_n, \alpha)\}$

$\{\square, (\blacktriangle, \rightarrow)\}$

Onde:

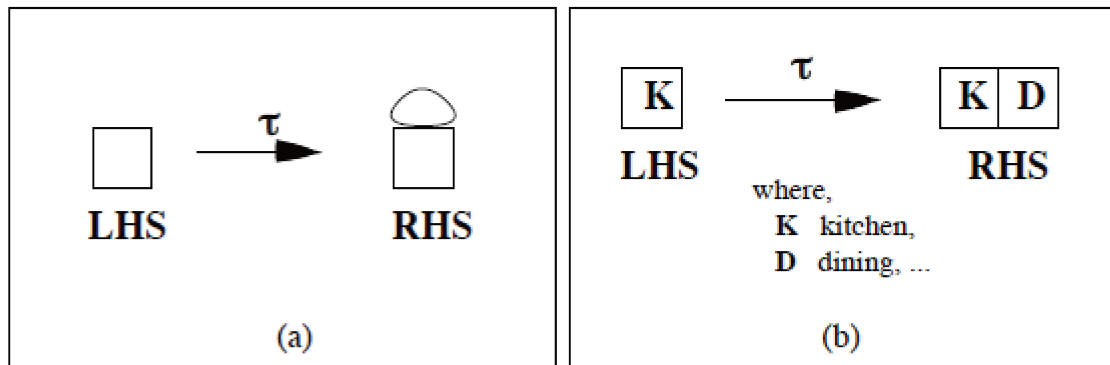
α é uma ação de transformação

E_n um novo elemento de design

E_x o elemento de design existente

S_r resultado da operação

Figura 53 – Sistema LHS - RHS



Fonte: “Representation and use of design knowledge in Evolutionary Design” (JO; GERO, 1995)

A gramática também operava com um sistema de marcadores onde, por exemplo, na existência de um elemento (E_x) com o marcador ($\bullet$), transforme-o introduzindo o elemento (E_n) com a ação α ,

$$\{ \{ (S_r, \text{fml}), (E_n, \alpha) \} \\ \{ (\square, *), (\square, \rightarrow) \} \}$$

Onde:

* marcador

O sistema foi montado com procedimentos que eram compostos por ações de conexão e ações de transformações topológicas, que eram representadas através de um sistema binário 00,01,10,11, ou seja, existiam quatro regras de conexão e quatro regras de transformações topológicas (JO; GERO, 1995):

$$(S_r \{ m, (L_n, ac, at) \}) \\ (S_g \{ L_n, ac, at \})$$

Onde:

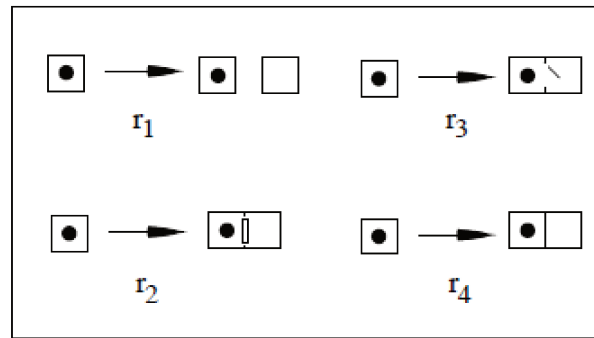
m marcador

L_n marcador para o novo elemento no espaço,

ac Ação de transformação de conectividade,

at Ação de transformação topológica

Figura 54 – Sistemas de conexão através de marcadores

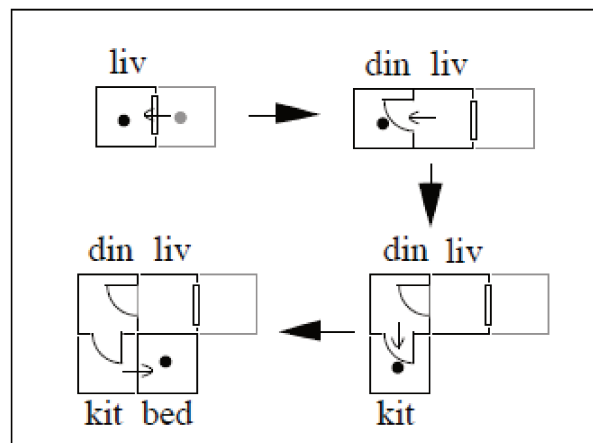


Fonte: “*Representation and use of design knowledge in Evolutionary Design*” (JO; GERO, 1995)

Onde:

- r1 sem relação,
- r2 com porta ou relação acessível,
- r3 janela ou relação através de transparência,
- r4 parede ou relação isolada.

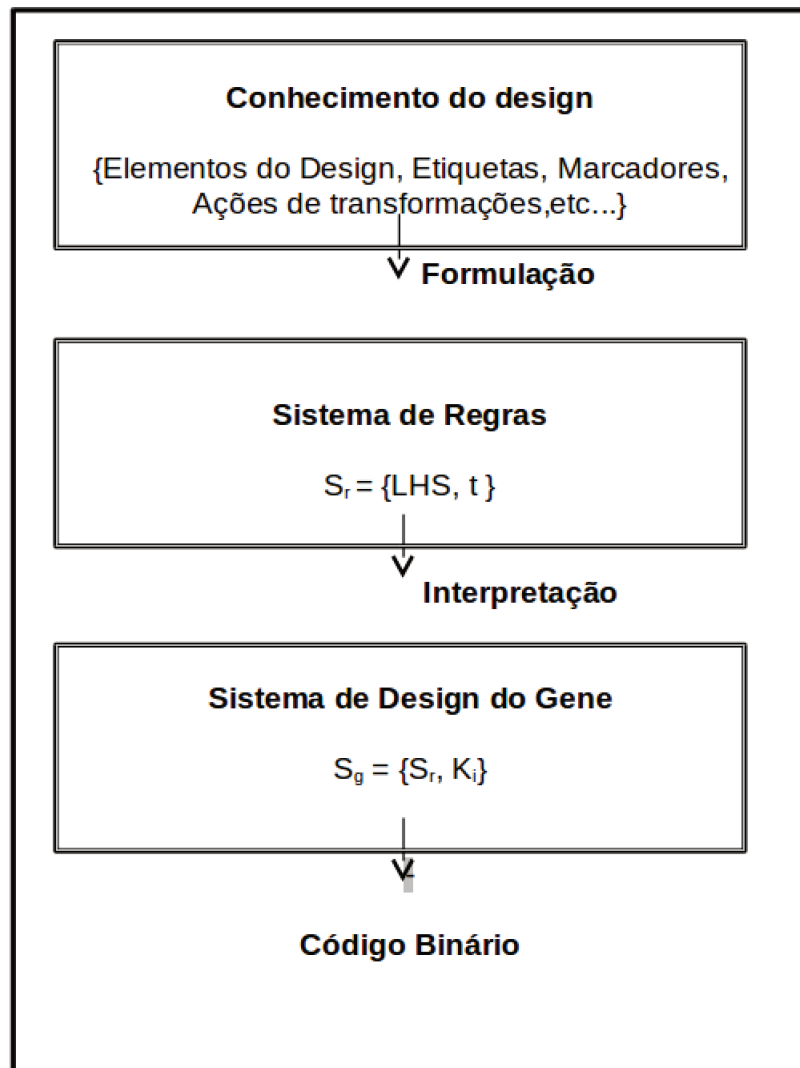
Figura 55 – Processo de conexão entre ambientes



Fonte: “*Representation and use of design knowledge in Evolutionary Design*” (JO; GERO, 1995)

Os ambientes eram também representados através de um sistema binário, em que: sala (00), sala de jantar (01), cozinha (10) e quarto (11), então o genótipo completo podia ser representado como: {...,(00, 10, 10)}, (01, 10, 10)}, (10, 01, 01)}, (11, 11, 00)}...}, ou 001010011010100101111100. O esquema geral da codificação do sistema e de sua decodificação em uma solução foi ilustrado através do fluxograma desenvolvido por Jo e Gero (1995):

Figura 56 – Fluxograma da codificação dos indivíduos realizado por Jo e Gero



Fonte: Adaptado de “Representation and use of design knowledge in Evolutionary Design” (JO; GERO, 1995)

A partir do sistema desenvolvido para representar os indivíduos de uma população, Jo e Gero desenvolveram um programa denominado *Evolutionary Design using Genetic Evolution* (EDGE), criado em uma plataforma UNIX utilizando linguagem C.

O algoritmo evolutivo seguia os seguintes passos: uma população inicial era gerada randomicamente e avaliada de acordo com sua aptidão, então um ou mais indivíduos eram selecionados e colocados no *mating pool* para passar por operações genéticas de recombinação e mutação. Os indivíduos transformados eram novamente avaliados e a nova população era composta por antigos e novos indivíduos, esse processo acontecia através de interações por várias gerações, até que a população convergisse ou um determinado número de gerações fosse alcançado. O software EDGE foi aplicado em um problema de *space planning* desenvolvido por Liggett e Mitchell (1981). O sistema era baseado em um

método heurístico de programação (método de *improvement*) com restrições geométricas e topológicas.

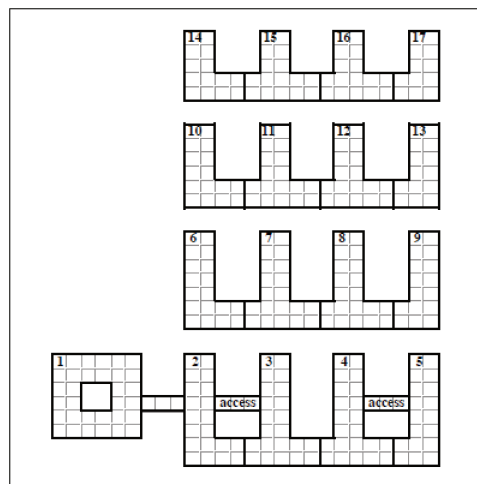
O problema consistia em um edifício de serviços de 4 andares (figura 57), composto por 21 escritórios, sendo que 5 deles eram previamente alocados no espaço. O problema de Ligget envolvia a otimização do custo de transporte entre pares de ambientes, bem como o tempo de percurso de um ambiente para outro. O método de avaliação utilizado era a típica equação de QAP ¹:

$$(\sum a_{ij} + \sum \sum q_{i1i2} c_{j1j2})$$

Onde:

a_{ij}	custo fixo de locação de um ambiente i para um ambiente j
c_{j1j2}	distancia de um local $j1$ para um local $j2$
i	unidade de ambiente a ser alocada
j	possível localização
q_{i1i2}	interações entre unidades de ambiente $i1$ e unidades de ambiente $i2$

Figura 57 – Edifício de serviços de 4 andares desenvolvido por Ligget em 1985



Fonte: “Representation and use of design knowledge in Evolutionary Design” (JO; GERO, 1995)

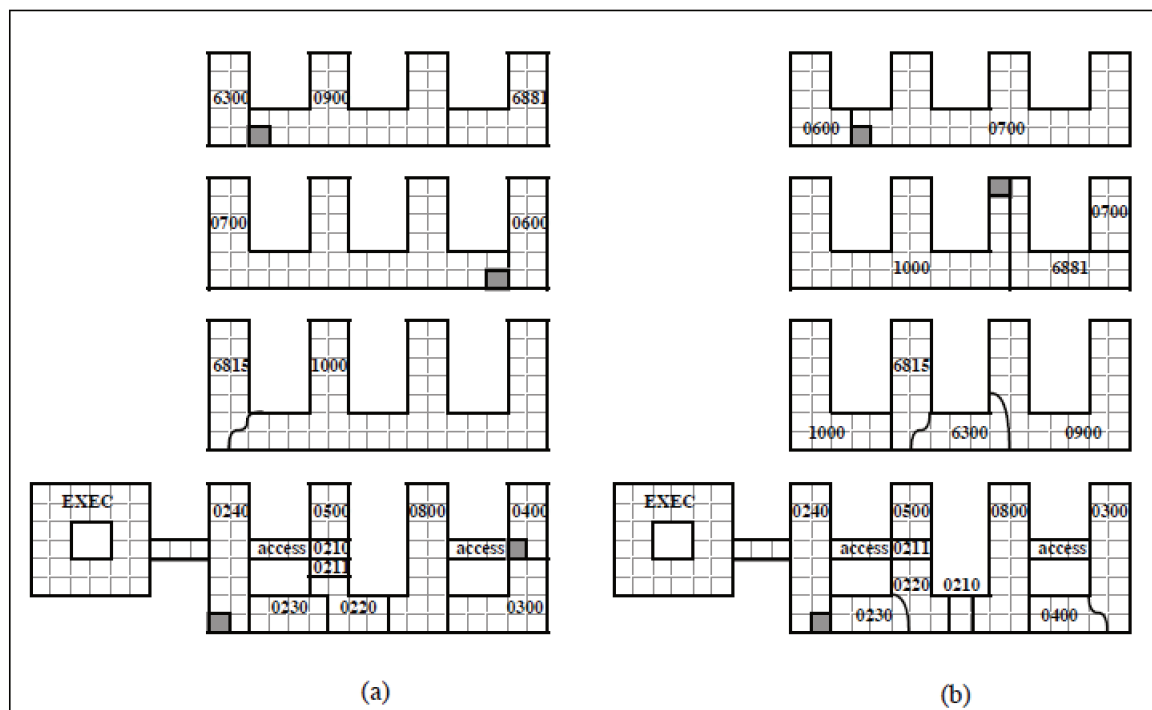
O genótipo dos indivíduos era calculado pela multiplicação do número de genes, do número de componentes do gene e do número de *bits* em um *byte*, ou seja, no caso do problema de Ligget, o comprimento do cromossomo que compunha a solução era 95 *bits*, sendo que 19 eram genes do design. Cada gene continha 1 componente e cada com-

¹ Equação de otimização que verifica o custo de alocação e do fluxo de materiais entre ambientes

ponente tinha 5 bits. Então cada gene era separado do genótipo e decodificado, de forma a corresponder às regras de design que mapeavam a solução, produzindo um fenótipo.

Após 500 gerações o comportamento produzido pelas soluções do sistema EDGE de Jo e Gero (1995) apresentou um melhoramento de 8,2 % da solução original produzida por Liggett e Mitchell (1981).

Figura 58 – (a) Solução produzida pelo método Liggett e (b) Solução produzida pelo método EDGE após 500 gerações



Fonte: “Representation and use of design knowledge in Evolutionary Design” (JO; GERO, 1995)

Jagielski e Gero (1997) também exploraram o problema de Liggett e Mitchell (1981) através de programação genética, porém esta abordagem adotou uma maneira diferente de representação dos cromossomos dos indivíduos, que foram representados através de uma cadeia de caracteres, em que cada caractere representava uma unidade do ambiente a ser alocado em uma matriz pré-definida (figura 59).

Li e Love (1998) desenvolveram, na linguagem *Visual Basic*, um software baseado em um AG para resolução de problemas de *One-to-One Assignment*, chamado *GenePlan*. Os cromossomos dos indivíduos eram representados através de números inteiros, e o método de avaliação usado foi o QAP. A figura 63 faz um resumo dos resultados obtidos por esses experimentos.

Figura 63 – Resumo dos resultados dos experimentos de Li e Love

Generation number (1)	Facility											Objection function value (13)
	2 (2)	3 (3)	4 (4)	5 (5)	6 (6)	7 (7)	8 (8)	9 (9)	10 (10)	11 (11)	12 (12)	
(a) Population size = 100												
0	5	2	3	4	9	6	8	1	11	7	10	18,168
10	2	8	11	3	7	5	6	1	9	4	10	17,162
20	7	9	3	2	5	6	4	1	8	11	10	17,013
30	4	2	6	11	7	9	8	1	5	3	10	16,857
40	2	8	7	9	5	4	6	1	11	3	10	16,742
50	6	9	8	11	5	3	2	1	7	4	10	16,586
60	5	11	8	6	3	9	2	1	4	7	10	16,443
70	8	4	7	6	11	5	2	1	9	3	10	16,268
80	7	4	6	11	2	9	5	1	8	3	10	16,022
90	11	5	8	7	2	9	3	1	6	4	10	15,090
(b) Population size = 70												
0	5	2	3	4	9	6	8	1	11	7	10	18,168
10	7	4	5	6	9	3	2	1	11	8	10	17,256
20	7	11	4	3	5	8	6	1	2	9	10	17,139
30	5	9	2	6	3	11	7	1	4	8	10	17,021
40	3	4	7	9	5	11	6	1	2	8	10	16,932
50	4	9	3	6	7	8	2	1	5	11	10	16,846
60	7	4	5	11	8	6	3	1	2	9	10	16,715
70	2	6	4	11	7	9	8	1	5	3	10	16,582
80	9	4	3	7	2	5	8	1	11	6	10	16,563
90	5	9	11	8	4	3	2	1	7	6	10	15,482
(c) Population size = 30												
0	5	2	3	4	9	6	8	1	11	7	10	18,168
10	6	5	4	2	7	11	9	1	8	3	10	17,130
20	6	8	11	4	9	5	7	1	3	2	10	17,041
30	11	3	5	2	7	6	8	1	9	4	10	16,784
40	5	7	4	11	8	6	3	1	9	2	10	16,768
50	2	3	5	9	4	11	7	1	8	6	10	16,639
60	6	4	8	5	9	11	2	1	7	3	10	16,438
70	8	11	2	9	6	4	3	1	5	7	10	16,412
80	9	2	3	7	8	5	6	1	4	11	10	16,325
90	6	8	2	7	3	9	4	1	5	11	10	16,320
(d) Population size = 50												
0	5	2	3	4	9	6	8	1	11	7	10	18,168
10	4	7	3	5	6	2	9	1	11	8	10	18,025
20	6	2	3	4	7	5	9	1	8	11	10	17,734
30	6	4	7	3	11	2	5	1	9	8	10	17,549
40	4	11	7	3	9	6	2	1	5	8	10	17,438
50	4	11	3	9	2	7	8	1	6	5	10	17,046
60	8	3	11	2	6	7	9	1	5	4	10	16,852
70	7	4	5	3	6	11	2	1	8	9	10	16,739
80	11	9	3	6	5	2	8	1	4	7	10	16,532
90	9	6	8	2	7	3	4	1	5	11	10	16,213
(e) Population size = 150												
0	5	2	3	4	9	6	8	1	11	7	10	18,168
10	4	11	5	2	7	6	8	1	9	3	10	17,638
20	5	6	7	9	3	2	8	1	4	11	10	16,735
30	11	9	3	6	5	2	8	1	4	7	10	16,532
40	11	2	5	6	8	3	7	1	4	9	10	16,470
50	9	7	2	5	3	11	8	1	4	6	10	16,103
60	11	9	2	7	5	6	8	1	4	3	10	16,089
70	11	8	3	4	2	9	6	1	7	5	10	15,948
80	9	3	4	6	7	11	5	1	8	2	10	15,872
90	8	5	6	9	7	11	3	1	4	2	10	15,656

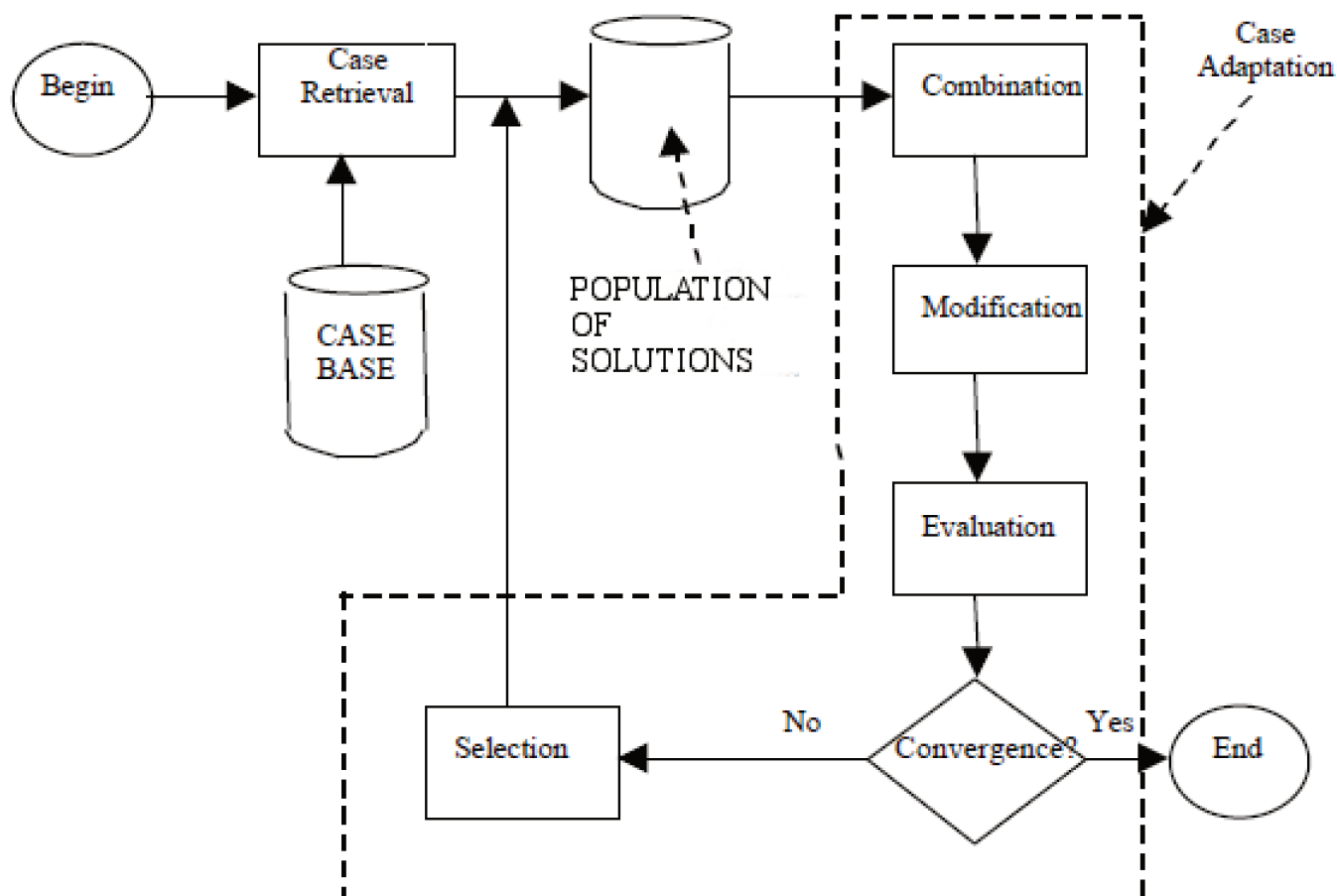
Fonte: "Site-level facilities layout using genetic algorithms" (LI; LOVE, 1998)

Kochhar *et al.* (1998) desenvolveram um algoritmo de GA denominado HOPE, em que o modelo de busca de solução também estava baseado no método de QAP. Tavakkoli-Moghaddain e Shayan (1998) utilizaram os mesmos métodos (AG + QAP) para resolução de problemas de *One-to-One Assignment*.

Garza e Maher utilizaram de maneira inusitada o modelo de AG com QAP para a resolução do problema de *One-to-One Assignment*. Criaram o algoritmo através de um

modelo de *case-based reasoning* (CBR), método que utiliza um banco de dados de soluções pré-definidas. Esse algoritmo CBR passava por um processo evolutivo através de uma sub-rotina, desenvolvida através de um AG (figura 64).

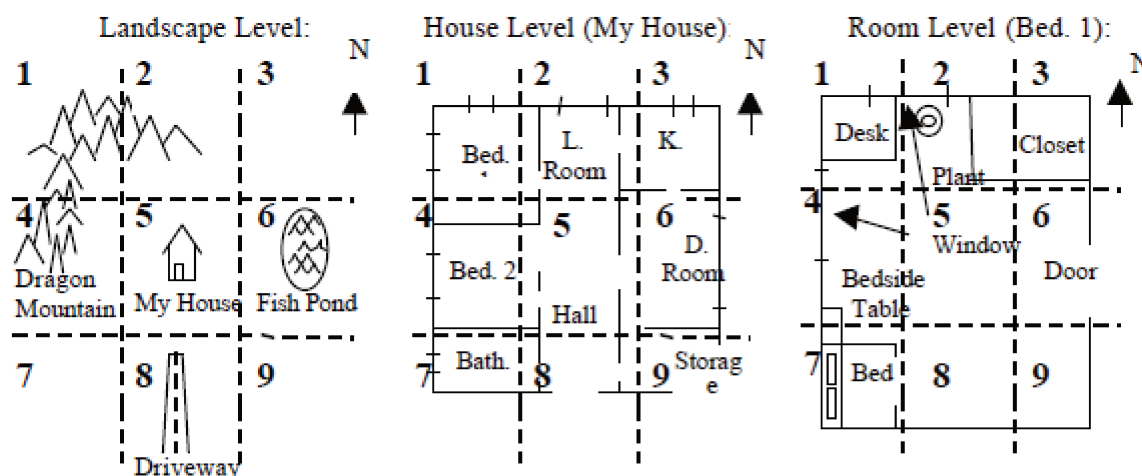
Figura 64 – Fluxograma do algoritmo desenvolvido por Garza e Maher



Fonte: *Evolving design layout cases to satisfy Feng Shui constraints* (GARZA; MAHER, 1999)

A ênfase do algoritmo estava em criar novas soluções, baseadas em um conhecimento já obtido de soluções passadas, pois as restrições da função objetivo estavam destinadas a otimizar questões subjetivas, ligadas à cultura chinesa do Feng-shui (figura 65).

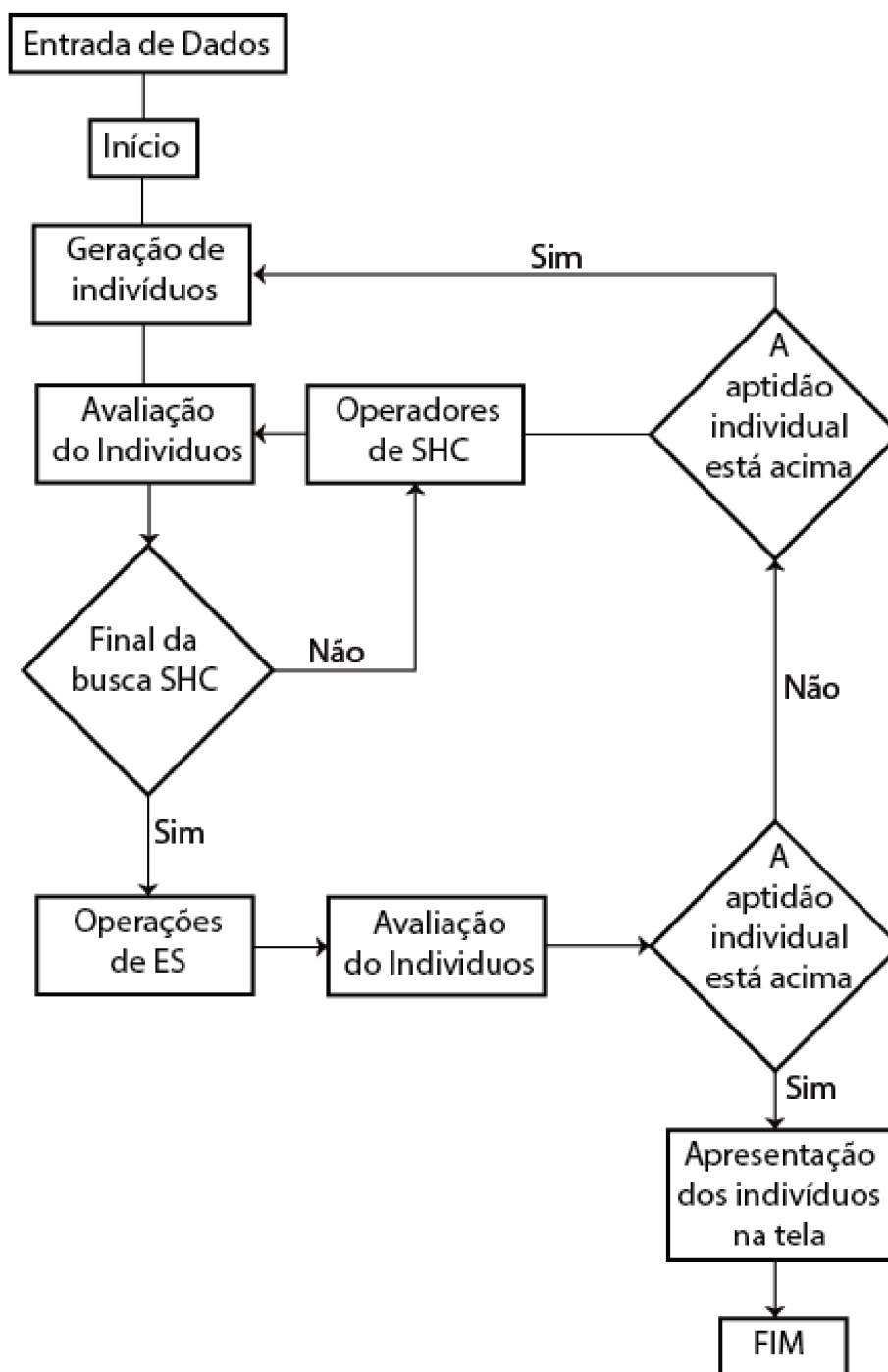
Figura 65 – Fluxograma do algoritmo desenvolvido por Garza e Maher



Fonte: “*Evolving design layout cases to satisfy Feng Shui constraints*” (GARZA; MAHER, 1999)

Rodrigues *et al.* (2013a) desenvolveram um algoritmo de estratégias evolutivas combinado com o método de *Stochastic Hill Climbing* (SHC). O algoritmo foi denominado de *Evolutionary Program for the Space Allocation Problem* (ESAP), e seu primeiro passo era gerar uma população randômica através do método de estratégia evolutiva, então os indivíduos eram avaliados através do processo de SHC e ranqueados de acordo com a função objetivo que avaliava 7 critérios: **conectividade e adjacência, sobreposição de ambientes, sobreposição de aberturas, orientação das aberturas, dimensão dos ambientes, compacidade dos ambientes e limites do espaço**. Os multicritérios eram avaliados através de uma somatória de pesos em que cada objetivo poderia ser avaliado de acordo com seu valor de importância (soma ponderada). O próximo passo do algoritmo era analisar o grupo de elite dos indivíduos das últimas 5 buscas, e caso fossem maior que o negativo da média das avaliações baseadas na somatória de pesos obtidos, o segundo estágio terminava e o algoritmo retornava para o primeiro passo da ES; caso contrário o algoritmo de SHC continuava aplicando uma série de operadores para cada indivíduo. Quando era encontrada a condição de termino do SHC, o segundo estágio terminava e o primeiro estágio era retomado. Os operadores de mutação da EE então geravam uma série de transformações geométricas nos indivíduos, que passavam novamente por uma avaliação, sendo ranqueados. Os que tivessem o valor de aptidão abaixo da média eram selecionados para a próxima geração, enquanto o restante era descartado e a reposição de indivíduos da população era refeita, através de novos indivíduos gerados randomicamente. Quando era encontrada a condição de termino da EE os melhores indivíduos eram mostrados. A figura 66 ilustra o processo através de um fluxograma elaborado por Rodrigues *et al.* (2013b).

Figura 66 – Fluxograma de funcionamento do algoritmo ESAP



Fonte: Adaptado de *An approach to the multi-level space allocation problem in architecture using a hybrid evolutionary technique* “ (RODRIGUES *et al.*, 2013a)

Rodrigues *et al.* (2013b) realizaram uma série de experimentos em edifícios de 1 ou mais pavimentos. O algoritmo se mostrou eficaz em alocar mais de 132 elementos, entre ambientes, portas, janelas, elevadores e escadas. O ESAP foi um avanço significativo nos métodos de SP com abordagem evolutiva, pela quantidade de elementos que foram alocados, pela otimização multicritério que avaliou uma série de quesitos e pela qualidade

dos resultados obtidos (figura 67).

Figura 67 – Resultado de um experimento multi-pavimento realizado por Rodrigues, Gaspar e Gomes

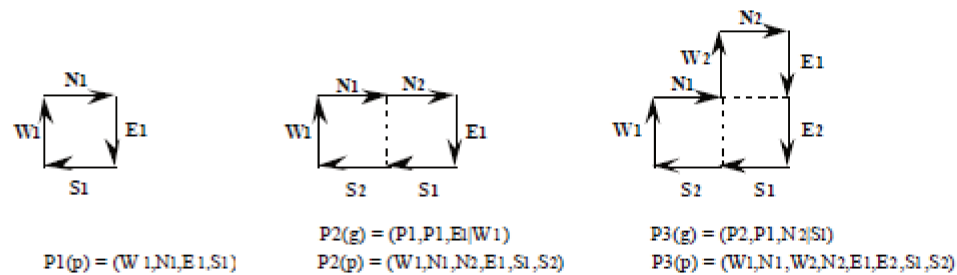


Fonte: "An approach to the multi-level space allocation problem in architecture using a hybrid evolutionary technique" (RODRIGUES *et al.*, 2013a)

4.4 Abordagem interativa da função objetivo

Baseados em uma gramática formal, Rosenman e Gero (1999) exploraram a criação de layouts através de regras simples e um processo evolutivo com Programação Genética implementado em C++. O cromossomo dos indivíduos era definido por vetores de orientação: norte, sul, leste e oeste (figura 68). O polígono então era criado a partir do somatório dos vetores obtidos pela operação.

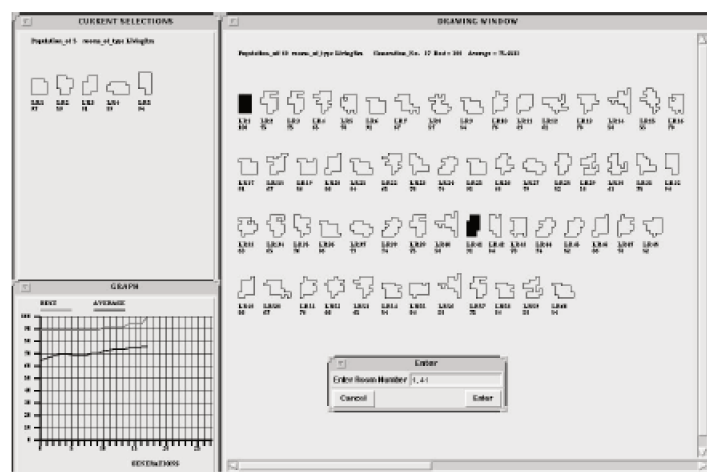
Figura 68 – Vetores de orientação constituindo ambientes



Fonte: "Evolving designs by generating useful complex gene structures" (ROSENMAN; GERO, 1999)

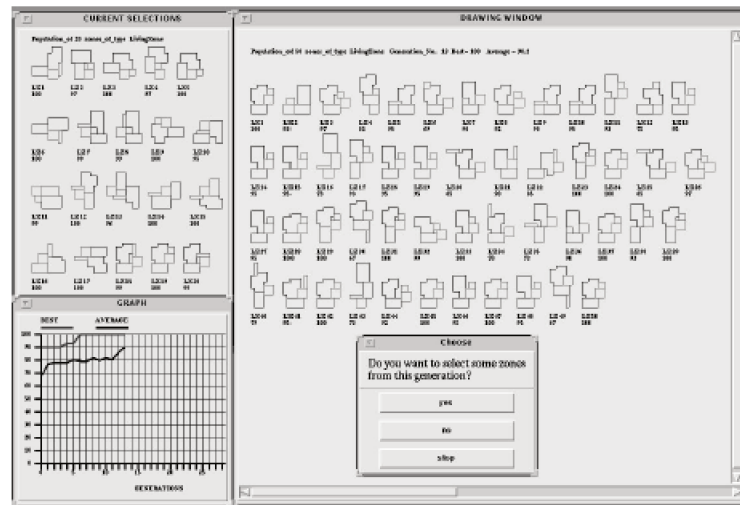
O processo de seleção dos indivíduos acontecia de maneira interativa com o usuário, e as soluções eram avaliadas por gerações e níveis hierárquicos: ambientes (figura 69), zonas (figura 70), e edificações (figura 71).

Figura 69 – População de ambientes gerados por Rosemman e Gero



Fonte: "Evolving designs By generating useful complex gene structures" (ROSENMAN; GERO, 1999)

Figura 70 – População de zonas geradas por Rosemman e Gero



Fonte: "Evolving designs by generating useful complex gene structures" (ROSENMAN; GERO, 1999)

Figura 71 – População de edificações geradas por Rosemman e Gero

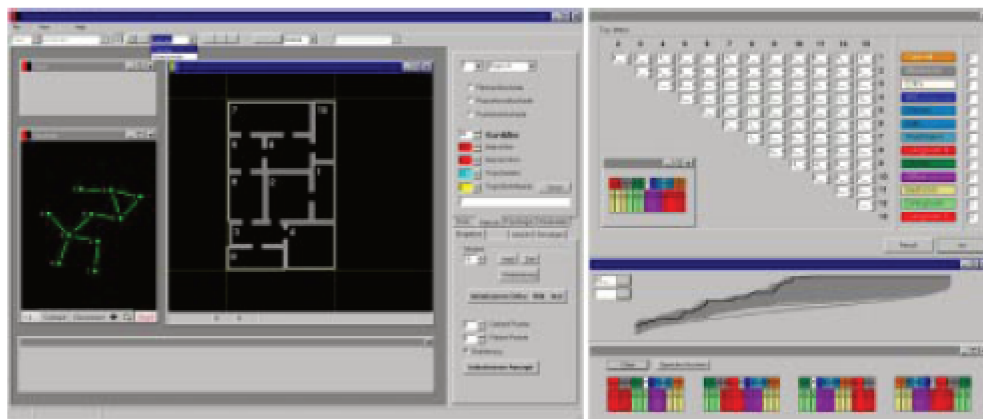


Fonte: "Evolving designs By generating useful complex gene structures" (ROSENMAN; GERO, 1999)

Elezkurtaj e Franck (2000) desenvolveram um algoritmo que combinava estratégias evolutivas e algoritmos genéticos. A função objetivo desse algoritmo estava especificada para solucionar questões de proporções dos ambientes e de suas relações com os vizinhos, através do método de QAP combinado ao *graph theory*, mas era o usuário final que determinava e selecionava as opções que melhor lhe atendesse. A função objetivo

funcionava de maneira híbrida entre interações com o usuário e restrições topológicas e geométricas dadas pelo algoritmo.

Figura 72 – Interface do programa desenvolvido por Elezkurtaj e Franck



Fonte: “*Geometry and Topology : A User-Interface to Artificial Evolution in Architectural Design*” (ELEZKURTAJ; FRANCK, 2000)

Harmanani *et al.* (2004) desenvolveram um algoritmo genético através do método de solução de *space planning* denominado *blocking*, em que polígonos representam ambientes com dimensões fixas, que devem ser combinados com outros ambientes em um dado espaço (LIGGETT, 2000) (figura 73). O método de avaliação da função objetivo foi o QAP. O algoritmo operava através de um sistema de computação paralela, em que as populações evoluíam através de um processamento paralelo.

Figura 73 – Método de blocking: (a) genótipo (b) fenótipo

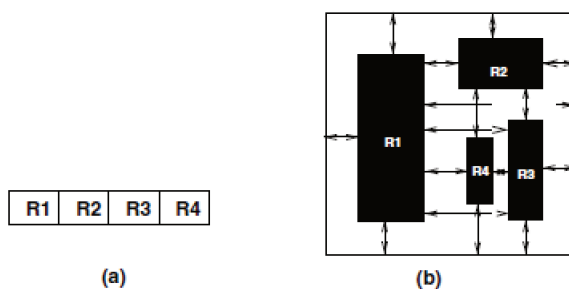


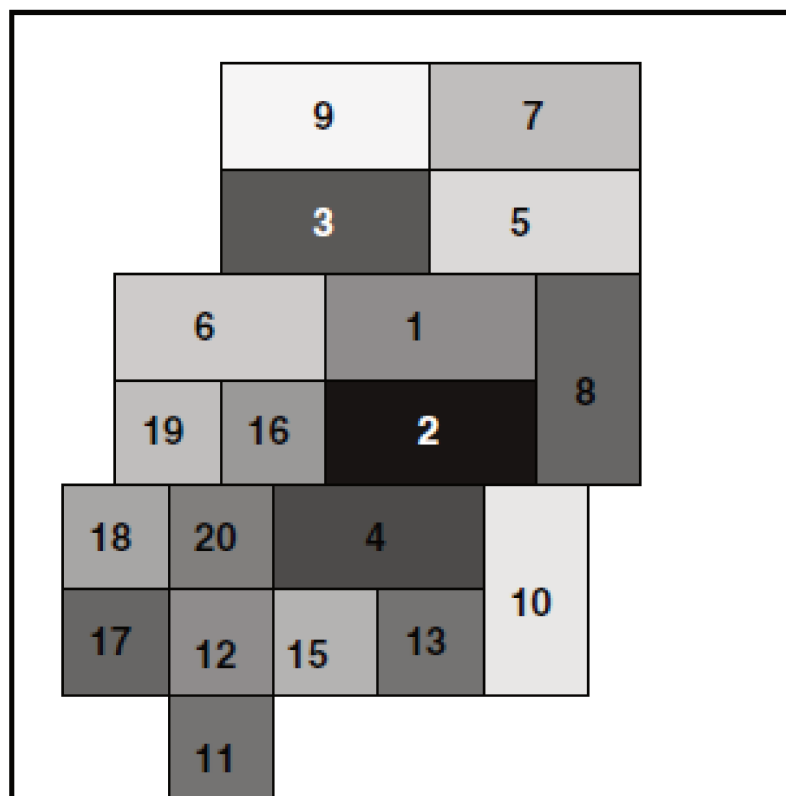
Fig. 1. (a) Chromosome representation, (b) corresponding site model.

Fonte: “*A parallel genetic algorithm for the geometrically constrained site layout problem with unequal-size facilities*” (HARMANANI *et al.*, 2004)

O programa foi desenvolvido em C++ em um *cluster* de máquinas paralelas no sistema operacional Linux. Os resultados obtidos foram comparados entre os experimen-

tos que utilizaram o algoritmo genético sequencial e outros experimentos que usaram o algoritmo paralelo. O resultado do experimento 4 pode ser visto na figura 74.

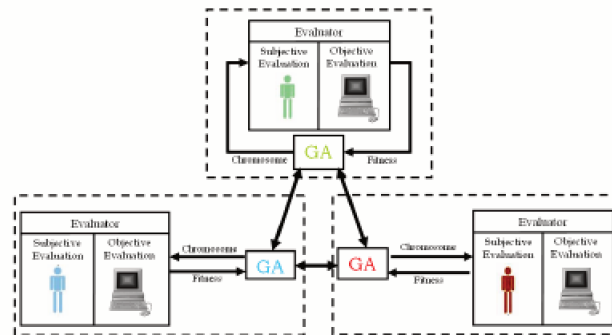
Figura 74 – Resultado do experimento 4



Fonte: “A parallel genetic algorithm for the geometrically constrained site layout problem with unequal-size facilities” (HARMANANI *et al.*, 2004)

Banerjee *et al.* (2008) propuseram um algoritmo genético para solução de problemas de *space planning* em que além de ter uma função objetivo interativa, essa interação era feita de maneira colaborativa por vários usuários.

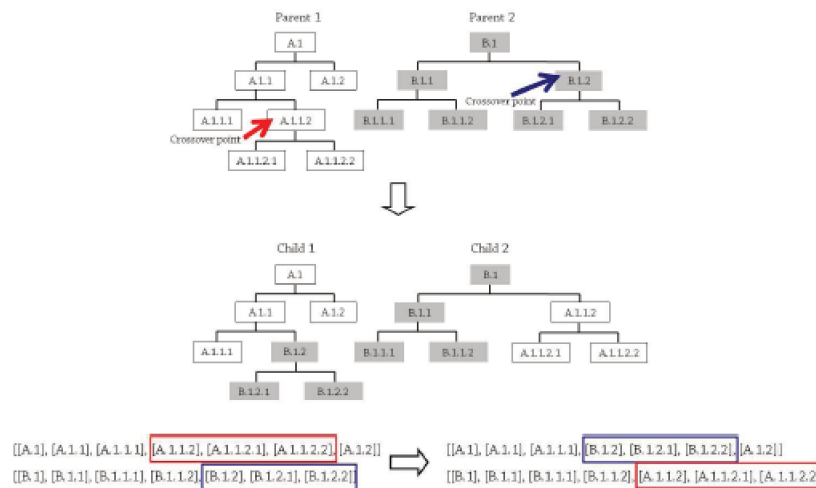
Figura 75 – Ilustração do funcionamento do processo de colaboração desenvolvido por Banerjee, Quiroz e Louis



Fonte: “A model of creative design using collaborative interactive genetic algorithms” (BANERJEE *et al.*, 2008)

O método de construção do *layout* utilizado foi o STS e programação genética (figura 76) - apesar de os autores identificarem o algoritmo como genético. A seleção dos indivíduos era feita de maneira *Peer-to-peer*.

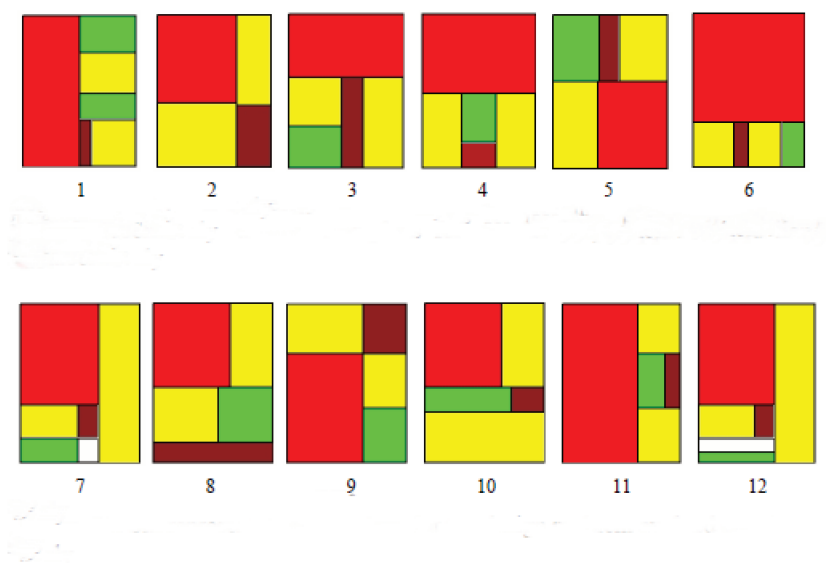
Figura 76 – Operação recombinação entre dois indivíduos da população



Fonte: “A model of creative design using collaborative interactive genetic algorithms” (BANERJEE *et al.*, 2008)

O processo colaborativo foi comparado com o método interativo, em que não havia colaboração, e os autores avaliaram que houve melhora das soluções nos cinco quesitos que avaliavam o aspecto prático dos *layouts* e a sua originalidade (figuras 77 e 78).

Figura 77 – Layout de 1 a 6 avaliados de forma não colaborativa e de 7 a 12 de maneira colaborativa



Fonte: “A model of creative design using collaborative interactive genetic algorithms” (BANERJEE *et al.*, 2008)

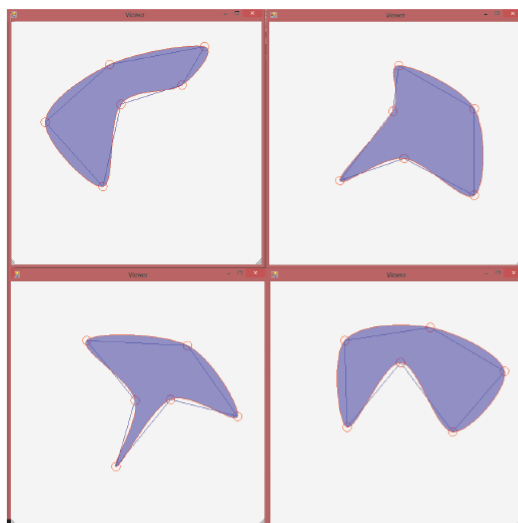
Figura 78 – Valores obtidos na avaliação dos *layouts*

DESIGN #	PRACTICALITY	ORIGINALITY	RANK
1	2.7	3.1	6
2	2.9	2.7	8
3	2.4	3.3	7
4	4.4	3.3	2
6	4.0	3.2	3
7	2.2	3.4	8
8	2.8	3.4	5
10	3.2	3.5	4
11	4.2	3.6	1
12	1.6	3.8	10

Fonte: Adaptado de “A model of creative design using collaborative interactive genetic algorithms” (BANERJEE *et al.*, 2008)

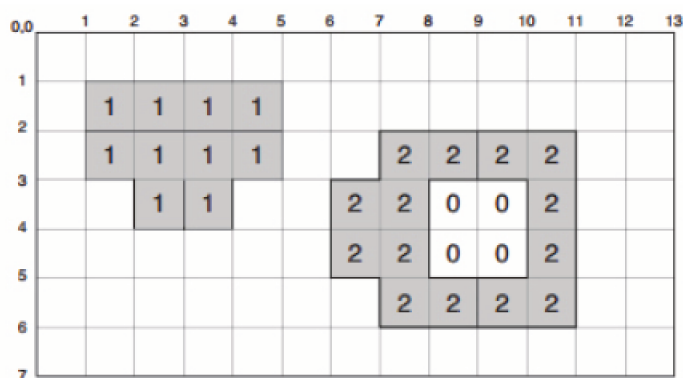
Fernando (2014) desenvolveu um algoritmo genético interativo para os estágios iniciais de concepção do layout arquitetônico. O usuário desenhava formas livres (figura 80), através de pontos de controle e o computador identificava esses desenhos através do fechamento da poligonal em um grid (figura 79). A função objetivo estava relacionada a questões de adjacência e dimensões das células do grid. As formas livres desenhadas a mão então evoluíam de acordo com os critérios estabelecidos pelo AG (figura 81).

Figura 79 – Formas livres desenhadas pelo usuário



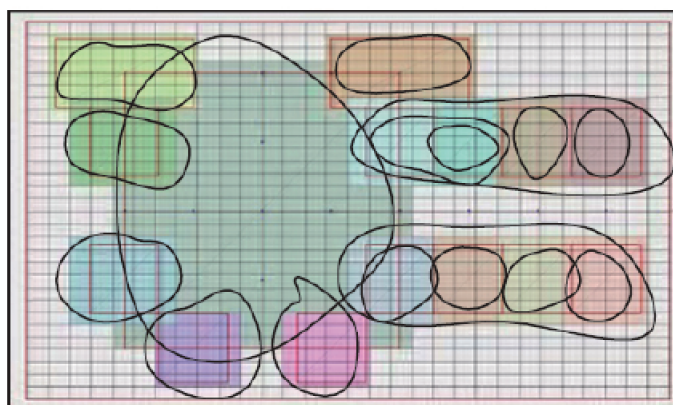
Fonte: “Space planning and preliminary design using artificial life” (FERNANDO, 2014)

Figura 80 – Adaptação das formas livres ao grid



Fonte: “Space planning and preliminary design using artificial life” (FERNANDO, 2014)

Figura 81 – Evolução das forma através do AG



Fonte: “Space planning and preliminary design using artificial life” (FERNANDO, 2014)

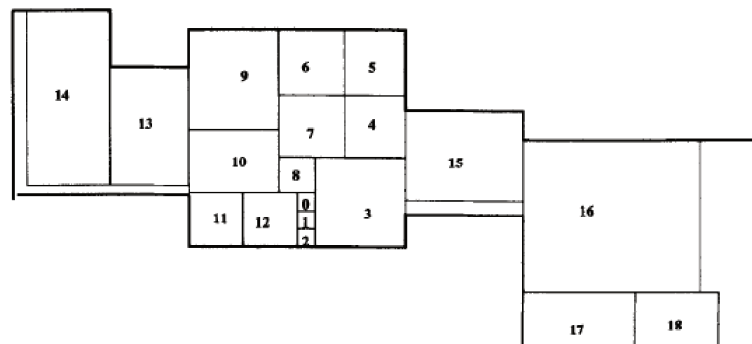
4.5 Engenharia Genética no aperfeiçoamento das soluções

A Engenharia genética (EG) aplicada a algoritmos evolutivos foi um método desenvolvido por Gero e Kazakov (1997a), que estendia o método clássico dos algoritmos genéticos através de um modelo baseado na engenharia genética aplicada a organismos naturais. O sistema de EG foi desenvolvido para identificar genes, sequências de genes ou estruturas de genes associados com o desempenho diante de uma função objetivo.

O método de EG identificava características do problema e automaticamente produzia cromossomos denominados “genes evoluídos”, que eram padrões de soluções introduzidas na população gerada randomicamente. Este processo acontecia antes dos passos de recombinação e mutação. Através desse método a convergência da população ocorria de maneira mais rápida e eficaz, pois identificava boas soluções recorrentes, preservando-as e reinserindo-as na população que havia sido gerada de maneira randômica.

Gero e Kazakov (1997a) testaram o método de EG em um problema de *space planning* composto por 19 ambientes, com a média da área das atividades de 30 m^2 . Os resultados demonstraram que, dos 19 ambientes, as boas soluções de alocação tendiam a se repetir, e que os ambientes de 0 a 12 formaram um gene evoluído, que consistia em um aglomerado de ambientes sem espaços vazios (figura 82).

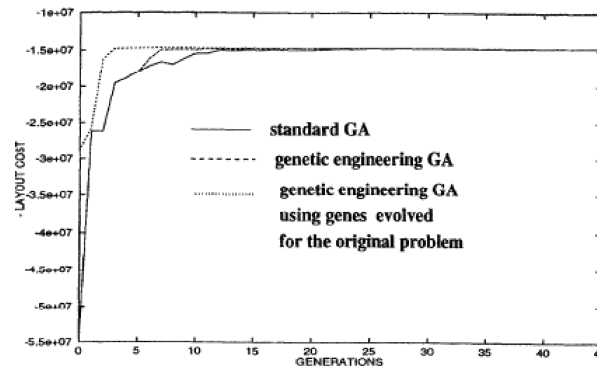
Figura 82 – Espaço de 0 - 12 tendiam a se aglomerar formando o que ocasionou em sua identificação como gene evoluído



Fonte: “*Evolving design genes in space layout planning problems*” (GERO; KAZAKOV, 1997a)

Gero e Kazakov (1997b), em uma outra pesquisa, realizaram um experimento com o problema de Liggett e Mitchell (1981), utilizando o método de engenharia genética. Eles compararam os resultados obtidos com o experimento realizado anteriormente por Jo e Gero (1995) (figura 83), e os resultados demonstraram que a utilização de genes evoluídos colaborou para a convergência da solução de maneira mais eficaz.

Figura 83 – Comparação entre a convergência da solução entre AG e AG com EG



Fonte: “*Learning and re-using information in space layout planning problems using genetic engineering*” (GERO; KAZAKOV, 1997b)

Homayouni (2007), em sua dissertação de mestrado, explorou os conceitos de engenharia genética aplicada a AG desenvolvidos por Gero e Kazakov (1997a), e implementou um algoritmo em *matlab* utilizando um *toolkit* de GA, ampliando-o com métodos de engenharia genética. O algoritmo de Houmayouni funcionava com dados de entrada através de perguntas feitas ao usuário. A primeira etapa de perguntas era:

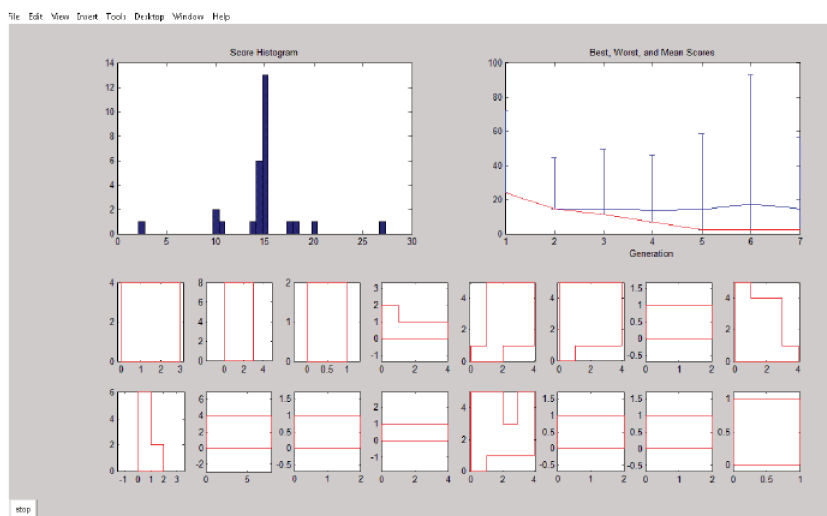
1. Quantos ambientes seu edifício possui?
2. Você quer minimizar a área do edifício ao máximo?
3. Você quer minimizar o perímetro do edifício?
4. Qual é a melhor dimensão (comprimento e largura) para o edifício?

Depois de selecionados esses dados o usuário confirmava e respondia a outras perguntas relacionadas ao primeiro ambiente do edifício:

1. Qual o nome do ambiente?
2. Qual a melhor área para o ambiente?
3. Você prefere formas compactas?
4. Qual a melhor proporção para o ambiente?

Após o usuário digitar os dados de entrada, as gerações do ambiente eram formadas, o processo iniciava as interações do algoritmo e todo o processo era apresentado através de uma interface, onde eram mostrados o histórico das melhores soluções e os indivíduos da população atual (figura 84).

Figura 84 – Sétima geração de um experimento realizado por Houmayouni



Fonte: “*A genetic algorithm approach to space layout planning optimization*” (HOMAYOUNI, 2007)

Depois de definidas todas as escolhas no nível do ambiente, o usuário determinava as relações de adjacências dos ambientes através de uma interface que apresentava uma matriz de adjacência (figura 85).

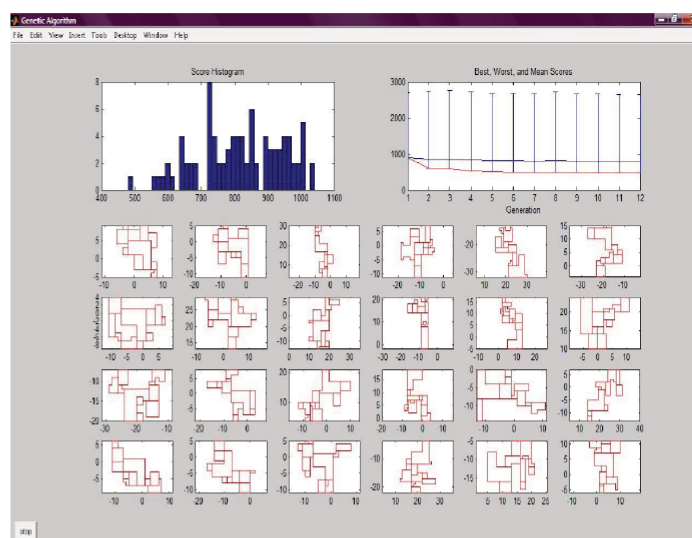
Figura 85 – Matriz de adjacência das relações entre os ambientes do algoritmo realizado por Houmayouni

The figure shows a web-based interface for specifying adjacency priorities. At the top, a yellow banner reads: 'Specify the adjacency priorities between the rooms by assigning values between 0 and 10 to them:'. Below this, there is a grid of colored boxes representing different rooms: outside, Bedroom, Livingroom, Kitchen, Diningroom, Bathroom, Master Bedroom, Bath, Laundry, Office, and Entrance. Each room is represented by a colored box with its name. Below the room names is a matrix of input fields where users can assign values between 0 and 10 to represent the adjacency priority between the rooms. A 'Submit' button is located at the bottom center of the interface.

Fonte: “*A genetic algorithm approach to space layout planning optimization*” (HOMAYOUNI, 2007)

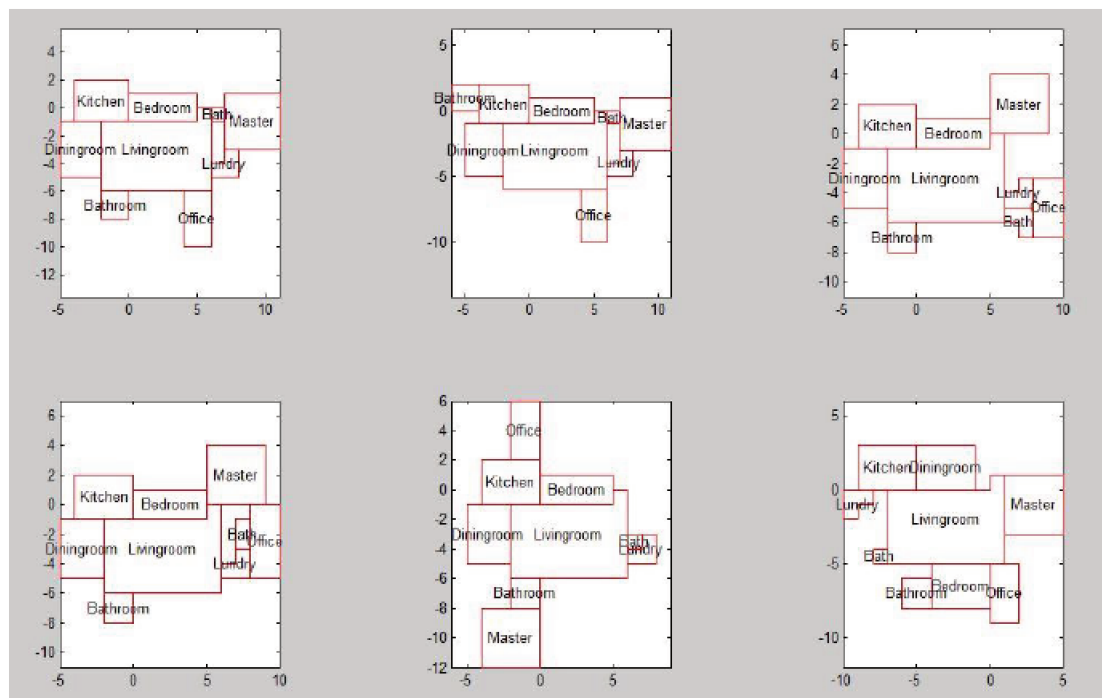
Após a definição das relações de adjacência o algoritmo operava na escala do edifício, evoluindo as melhores soluções de *layout*.

Figura 86 – Interações realizadas na escala do edifício do algoritmo de Houmayouni



Fonte: “A genetic algorithm approach to space layout planning optimization” (HOMAYOUNI, 2007)

Figura 87 – Resultados dos layouts obtidos através de um experimento realizado por Houmayouni



Fonte: “A genetic algorithm approach to space layout planning optimization” (HOMAYOUNI, 2007)

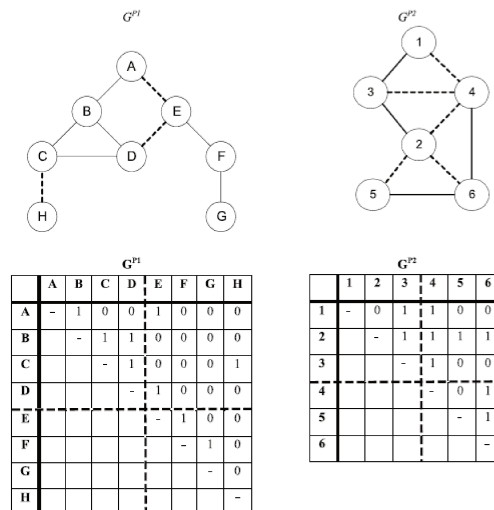
4.6 Graph Theory e o método evolutivo

Wong e Chan (2009) desenvolveram um algoritmo evolutivo denominado EvoArch, que tinha como objetivo encontrar, através de grafos, as melhores situações topológicas entre ambientes. O algoritmo seguia os passos abaixo:

1. Inicializava uma população randômica de grafos e codificava esses grafos através de uma matriz de adjacências.
2. Avaliava os grafos.
3. Selecionava dois grafos para reproduzir através do método de roleta.
4. Aplicava os operadores de recombinação e mutação nos grafos escolhidos.
5. Trocava os piores indivíduos da população pelos novos descendentes.
6. Repetia os passos 2 a 5 até que a função de conclusão (que estabelece o término da busca do algoritmo por soluções) fosse encontrada.

O processo de recombinação dos grafos acontecia na matriz de adjacência, através de um ponto de corte, como demonstrado na figura 88.

Figura 88 – Processo de recombinação através da matriz de adjacência



Fonte: “EvoArch: An evolutionary algorithm for architectural layout design ” (WONG; CHAN, 2009)

A função objetivo estava baseada em um processo de avaliação denominado *Adjacency Preference Scale* (APS) 89, em que se ranqueavam, através de uma matriz, as relações de preferência entre ambientes, então o objetivo era organizar essas relações de modo a maximizar o número obtido do somatório desse ranqueamento.

Figura 89 – *Adjacency Preference Scale* (APS) ranqueamento dos ambientes

	1 SA	2 MA	3 BED	4 LR	5 D&K	6 B	7 CP	8 P	9 CIR	10 EXT
1. Study area (SA)	-2	2	1	-2	-2	-2	-2	-1	3	3
2. Master ensuite (ME)	2	-2	2	-2	-2	-2	-2	3	3	3
3. Bedroom (BED)	1	2	2	-2	-2	3	-2	1	3	3
4. Living room (LR)	-2	-2	-2	-2	1	2	1	3	3	3
5. Dining & kitchen (D&K)	-2	-2	-2	1	-2	2	-2	1	3	2
6. Bathroom (B)	-2	-2	3	2	2	3	-2	-2	3	1
7. Car park (CP)	-2	-2	-2	1	-2	-2	-2	-2	3	3
8. Patio (P)	-1	3	1	3	1	-2	-2	-2	-2	3
9. Hall/stair/circulation area (CIR)	3	3	3	3	3	3	3	-2	3	-2
10. Exterior (EXT)	3	3	3	3	2	1	3	3	-2	3
Average	-0.2	0.3	0.9	0.5	-0.1	0.6	-0.7	0.2	2	2.2

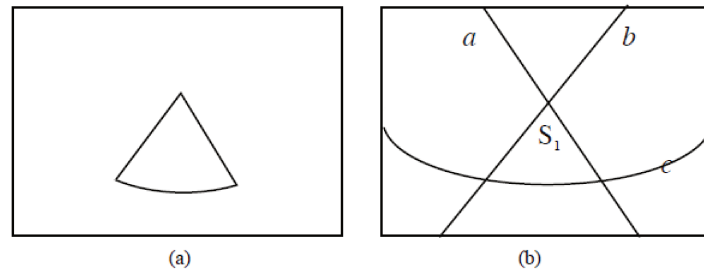
Fonte: “*EvoArch: An evolutionary algorithm for architectural layout design* ” (WONG; CHAN, 2009)

Thakur *et al.* (2010) criaram um algoritmo genético de resolução de problemas de *space planning* através de grafos e matrizes de adjacências. A matriz 4x4 tinha um limite de 16 ambientes e a função objetivo resolvia restrições topológicas e geométricas, além de uma sub-rotina que analisava as melhores rotas de fuga através de um algoritmo de Dijkstra.

4.7 Half plan e Kd Trees com abordagem evolutiva

Damski e Gero (1997) desenvolveram um sistema de resolução de problemas de *space planning* com abordagem evolutiva através de um método denominado *half plan*. Esse sistema subdividia um plano em sub-planos, através da identificação de bordas, gerando ambientes através de operadores lógicos booleanos (figura 90).

Figura 90 – (a) espaço (b) *half plan* que define o espaço



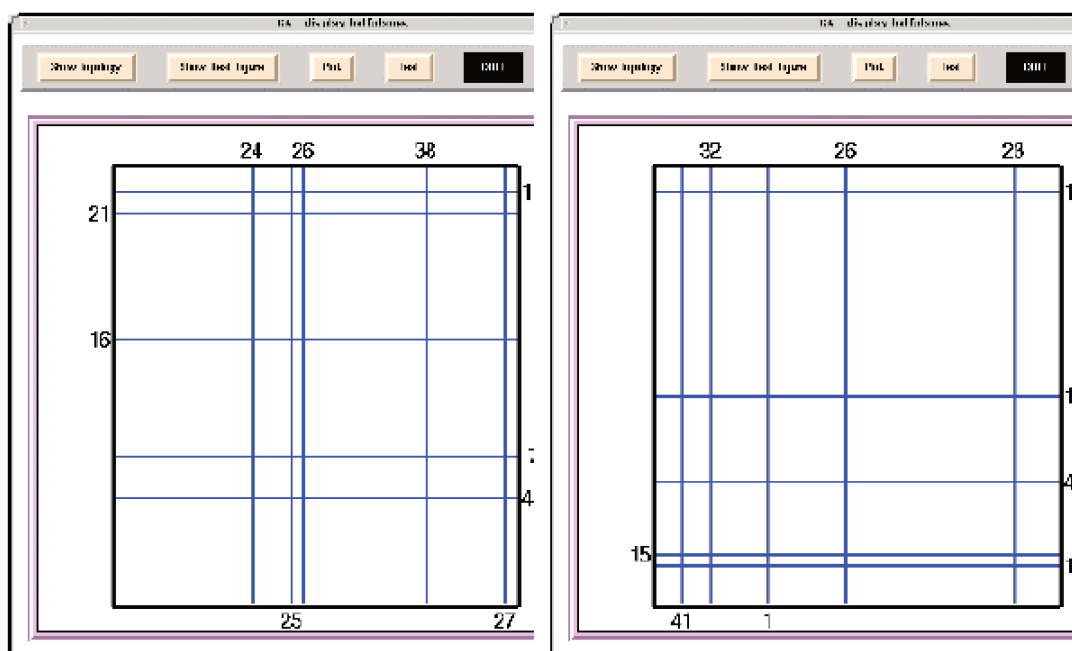
Fonte: “An evolutionary approach to generating constraint-based space layout topologies ” (DAMSKI; GERO, 1997)

As etapas de criação do sistema evolutivo baseado em *half-planes* foram definidas por Damski e Gero (1997):

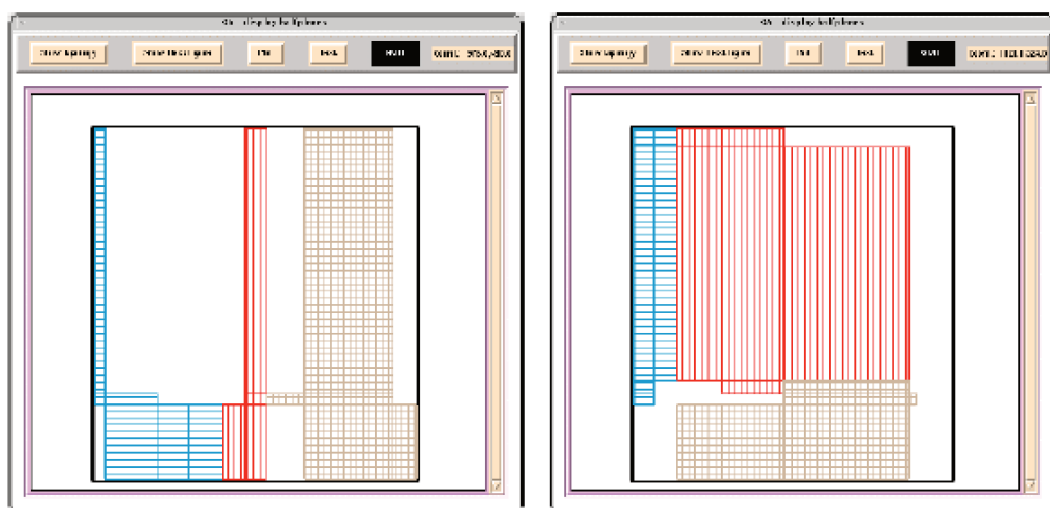
1. Uma lista de H (*half-planes*), é gerada
2. Uma população de arranjos de *half-planes* é gerada, onde cada membro H_i é uma sub lista de H.
3. Para cada H_i se gera uma lista de *layouts* L
4. Cada *layout* L é avaliado de acordo com uma série de restrições
5. Os *layouts* evoluem de acordo com o resultado da função objetivo
6. Cada arranjo de H_i é evoluído de acordo com a avaliação da população de *layouts* na topologia atual.
7. Arranjos são evoluídos em H.

O sistema foi desenvolvido de maneira hierárquica, sendo que em um primeiro momento são evoluídos os *half-planes* (figura 91), e em um segundo momento os *layouts* (figura 92), onde é utilizado um sistema multi-critério de avaliação da função objetivo, através de um método de otimização por Pareto.

Figura 91 – Processo evolutivo dos Half Planes realizado por Dasmki e Gero



Fonte: “*An evolutionary approach to generating constraint-based space layout topologies*” (DAMSKI; GERO, 1997)

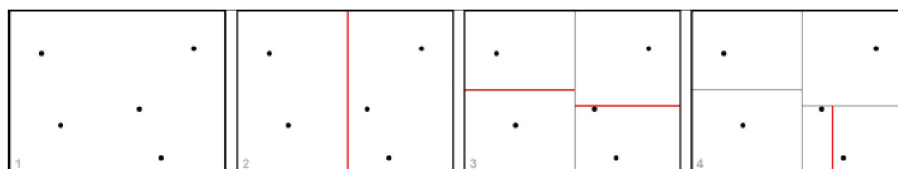
Figura 92 – Processo evolutivo dos *layouts* realizado por Dasmki e Gero

Fonte: “*An evolutionary approach to generating constraint-based space layout topologies*” (DAMSKI; GERO, 1997)

Knecht e Koenig (2010) desenvolveu um algoritmo genético para a criação de *layouts* baseado em um método de subdivisões do espaço denominado *KD-trees*. A sub-

divisão do espaço é obtida através de uma lista de pontos, onde são criados planos de partição na média obtida das coordenadas desses pontos (figura 93).

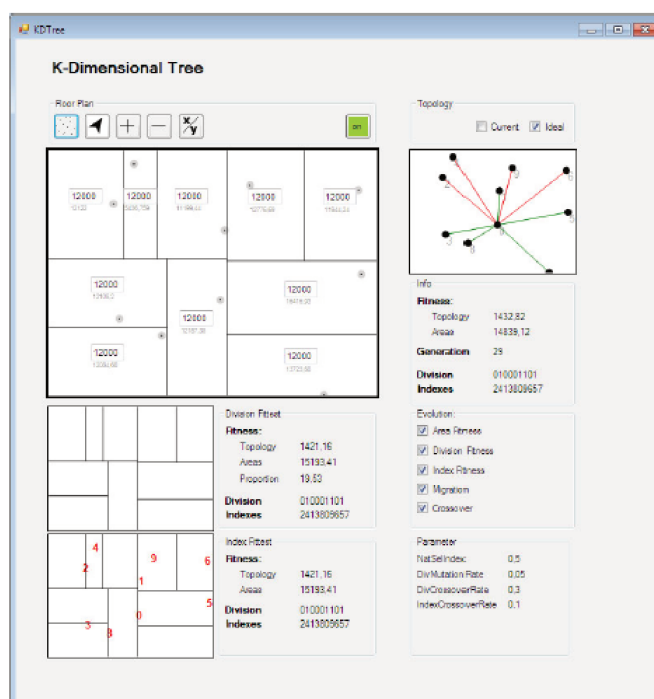
Figura 93 – Processo de decomposição do espaço através de KD-trees



Fonte: “*Generating floor plan layouts with k-d trees and evolutionary algorithms*” (KNECHT; KOENIG, 2010)

O método de avaliação utilizado foi multi-critério (figura 94).

Figura 94 – Interface do processo de otimização definido por Knecht

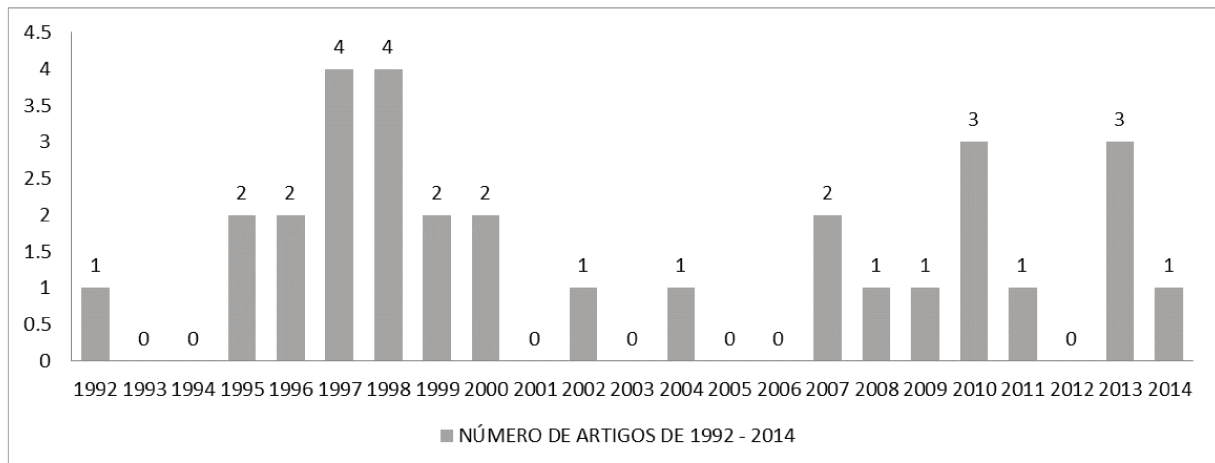


Fonte: “*Generating floor plan layouts with k-d trees and evolutionary algorithms*” (KNECHT; KOENIG, 2010)

4.8 Resultados da Revisão de literatura

Nesta seção iremos comparar os dados dos artigos selecionados através de gráficos. Como está demonstrado na figura 95, o período entre 1995 a 2000 foi o mais produtivo em publicações de artigos. Foram publicados ao menos 2 artigos por ano relacionados ao tema.

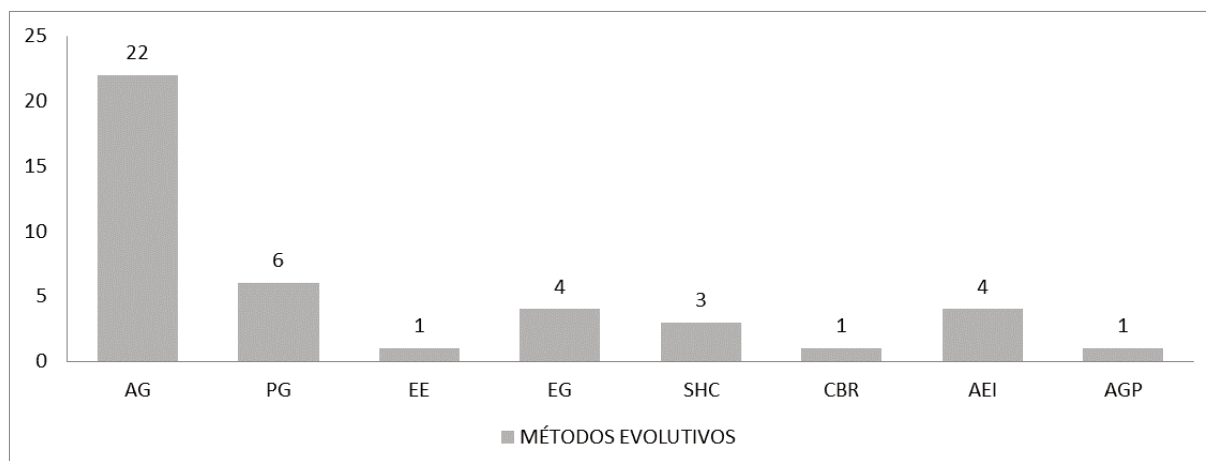
Figura 95 – Número de artigos publicados por ano



Fonte: Elaborado pelo autor

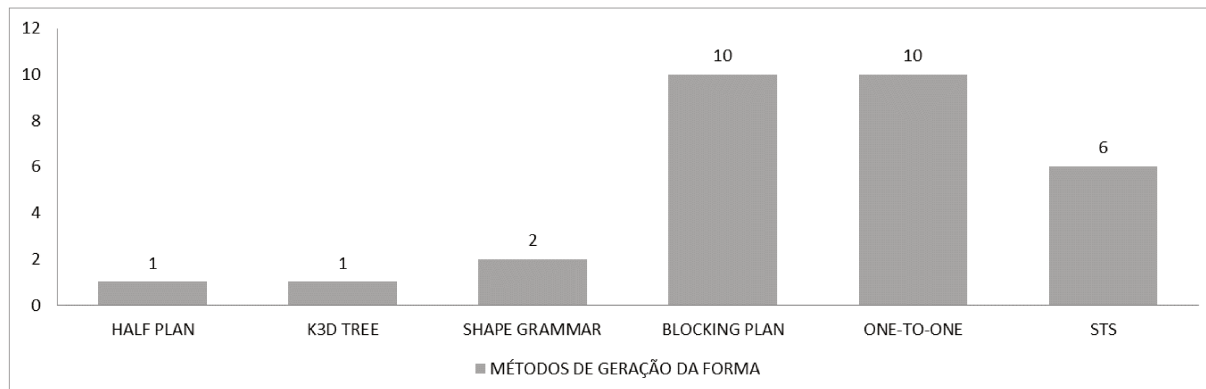
Diversos métodos evolutivos foram identificados nos artigos analisados, demonstrando a diversidade dos problemas relacionados ao tema. Foram identificados 6 métodos evolutivos, e dois métodos complementares aos evolutivos: (1) algoritmos genéticos (AG), (2) programação genética (PG), (3) estratégias evolutivas, (4) engenharia genética (EG), (5) algoritmo evolutivo iterativo (AEI), (6) algoritmo genético paralelo (AGP). Os dois métodos complementares aos evolutivos foram: (1) *Stochastic Hill Climbing* (SHC) e (2) *case-based reasoning* (CBR). A figura 96 apresenta o número de artigos publicados de acordo com o método evolutivo.

Figura 96 – Métodos evolutivos e complementares identificados nos artigos selecionados



Fonte: Elaborado pelo autor

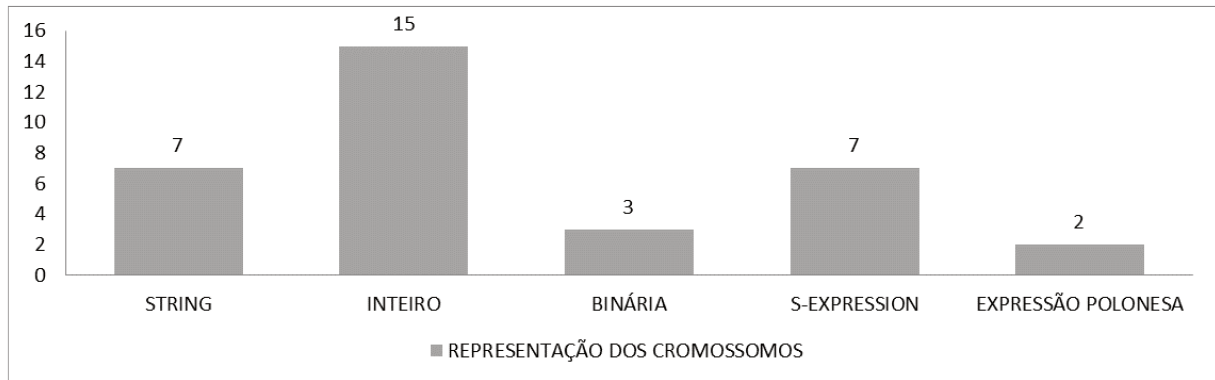
Entre os métodos de geração da forma dos *layouts*, *blocking* e *one-to-one assignment* foram os mais utilizados, seguidos pelo método de *slice tree structure* (figura 97).

Figura 97 – Métodos de concepção do *layout* identificados nos artigos selecionados.

Fonte: Elaborado pelo autor

Dentre os métodos utilizados para representar os cromossomos dos indivíduos, os números inteiros foram os mais utilizados (figura 98)

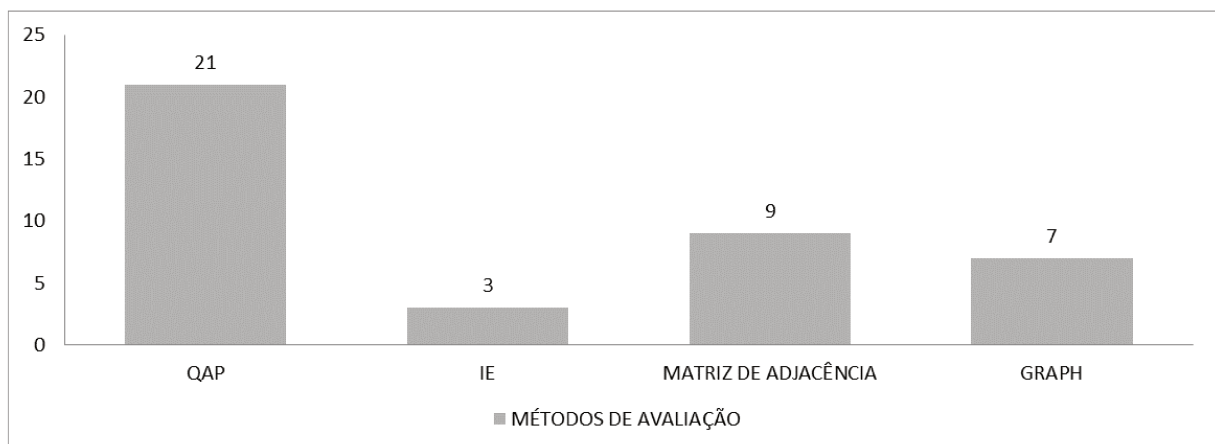
Figura 98 – Métodos de representação do cromossomo identificados nos artigos selecionados



Fonte: Elaborado pelo autor

O método de avaliação mais utilizado nos algoritmos evolutivos foi o QAP. Nota-se que a equação formulada pelo método de QAP facilita a transcrição para a função objetivo do algoritmo evolutivo.

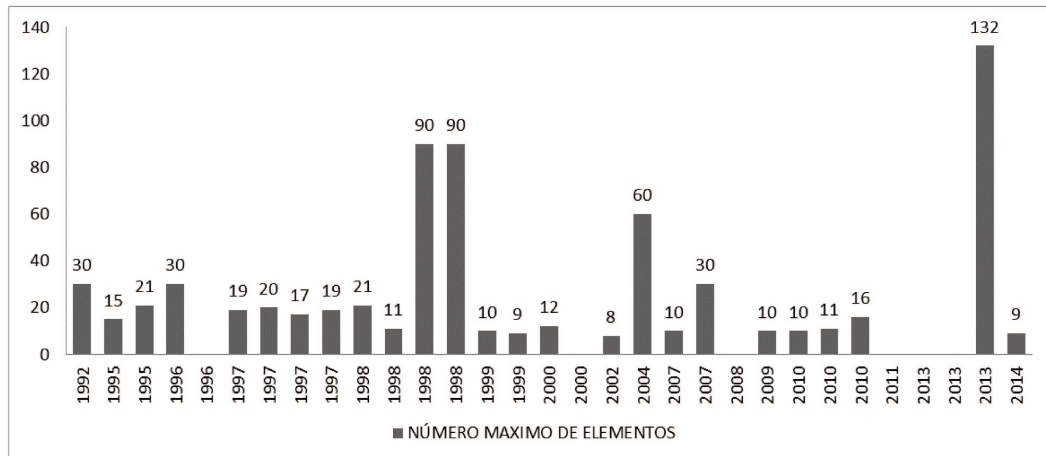
Figura 99 – Métodos de avaliação utilizados nos artigos selecionados



Fonte: Elaborado pelo autor

Gero e Kazakov (1997a) argumentaram que o tempo computacional é demasiadamente alto para projetos de larga escala em problemas de space planning. O avanço da computação nos últimos anos aparentemente não impactou significativamente o desenvolvimento desse tipo de problema. Identifica-se apenas um salto significativo no número de ambientes alocados em dois artigos publicados (RODRIGUES *et al.*, 2013a; RODRIGUES *et al.*, 2013b). A figura 100 ilustra o maior número de ambientes computados nos artigos analisados.

Figura 100 – Número máximo de ambientes computados nos artigos analisados, por ano

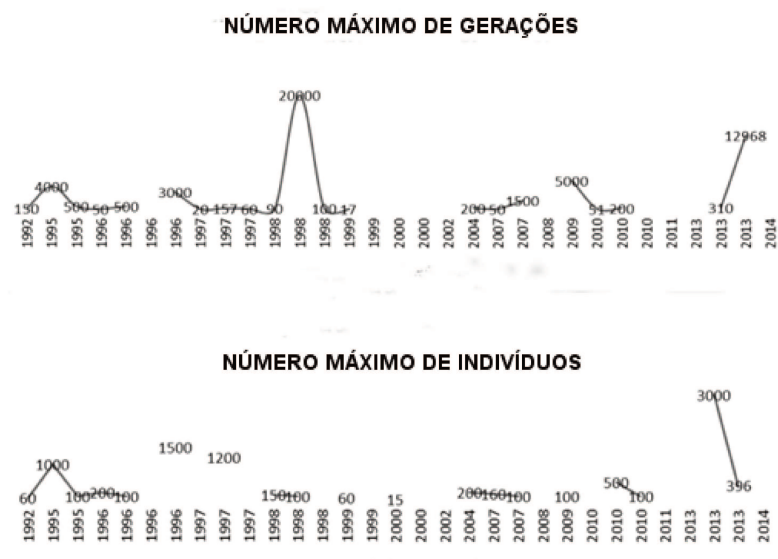


Fonte: Elaborado pelo autor

Em 15 anos, de 1998 a 2013, houve um acréscimo de 42 ambientes nos *layouts* computados, porém esse crescimento não foi linear ou exponencial, como tem sido no caso do desenvolvimento computacional.

Comparando-se o número máximo de elementos por população e o número máximo de gerações (interações do algoritmo), nesse período, verifica-se que foram necessárias 7000 interações a mais por população máxima em 1998 do que em 2013, sendo que o número de elementos computados nos artigos analisados de 1998 era 2850 vezes menor do que em 2013.

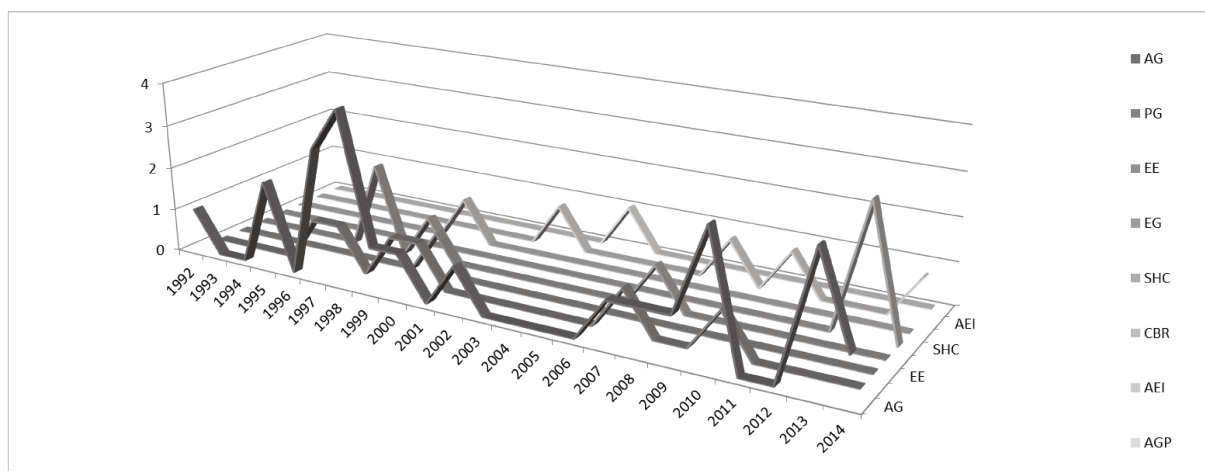
Figura 101 – Número máximo de gerações nos artigos analisados por ano e número máximo da população por ano respectivamente.



Fonte: Elaborado pelo autor

Os estudos demonstraram a preferência pelo uso de algoritmos genéticos ou pela hibridização dos métodos para a resolução de problemas de *space planning* durante o período estudado. A figura 102 mapeia os métodos evolutivos durante o período.

Figura 102 – Número máximo de gerações nos artigos analisados por ano e número máximo da população por ano respectivamente.



Fonte: Elaborado pelo autor

4.8.1 Demandas para trabalhos futuros

Foram checados os prospectos e as demandas para futuros trabalhos relatados nos artigos analisados. Identificamos como principais demandas: (1) refinamento dos métodos de recombinação (KOCHHAR *et al.*, 1998), (2) mais aplicações que se utilizem de uma função objetivo interativa (HOMAYOUNI, 2007), (3) mais métodos de engenharia genética que aperfeiçoem os operadores de recombinação, (4) mais aplicações relacionadas a problemas tridimensionais de layout através de uma única função objetivo (HOMAYOUNI, 2007; DOULGERAKIS, 2007; FLACK; ROSS, 2011) e (5) mais aplicações relacionadas a ambientes não ortogonais (KNECHT; KOENIG, 2010). Todas essas demandas apontam para futuros trabalhos e não foram exploradas na totalidade de suas possibilidades até o momento.

4.8.2 Considerações sobre *space planning* com abordagem evolutiva

Os métodos de SP com abordagem evolutiva vêm sendo estudados a mais de 20 anos, sendo ainda um campo com muitas possibilidades não exploradas de pesquisa. O avanço da computação nos últimos anos aparentemente não impactou significativamente o desenvolvimento de resoluções para o tema. Sendo assim, faz-se necessário o desenvolvimento de novos algoritmos que otimizem o processamento dos computadores, permitindo o aumento do número de ambientes computados.

No campo do projeto computacional, os métodos de SP são explorados desde o surgimento dessa temática. Entretanto, novos métodos devem ser criados, e os antigos métodos aperfeiçoados, para que esse campo de pesquisa continue se desenvolvendo.

Apesar dos novos ambientes computacionais terem colaborado para a popularização do campo do design computacional (MARTINO, 2015), poucos algoritmos foram desenvolvidos voltados à resolução de problemas de *space planning*.

Neste trabalho propomos alguns algoritmos experimentais em *Grasshopper*, que serão voltados a atender problemas de *space planning* com abordagem evolutiva. O objetivo destes algoritmos é dar um passo inicial para a popularização de alguns desses métodos através de uma interface amigável aos arquitetos.

No próximo capítulo iremos discorrer sobre os métodos e materiais que foram utilizados nos experimentos.

Parte IV

Métodos e Materiais

5 Métodos e Materiais

Nesse capítulo serão descritos os métodos e materiais utilizados para a realização dos experimentos deste trabalho. Para a execução desta pesquisa, as etapas abaixo discriminadas foram seguidas, com os métodos e técnicas relacionados:

5.1 Materiais

5.1.1 Especificações de Hardware e Software

Hardware

Para a implementação dos experimentos desta pesquisa foi utilizado um laptop ASUS com as seguintes especificações:

Modelo : ASUS S46CM

Processador : Intel core i7 3517U

Sistema Operacional : Windows 8

Memoria Ram: DDR3 1600 MHZ SDRAM 8 GB

Placa de Video: NVIDIA GeForce GT 635m 2 GB DDR3 VRAM

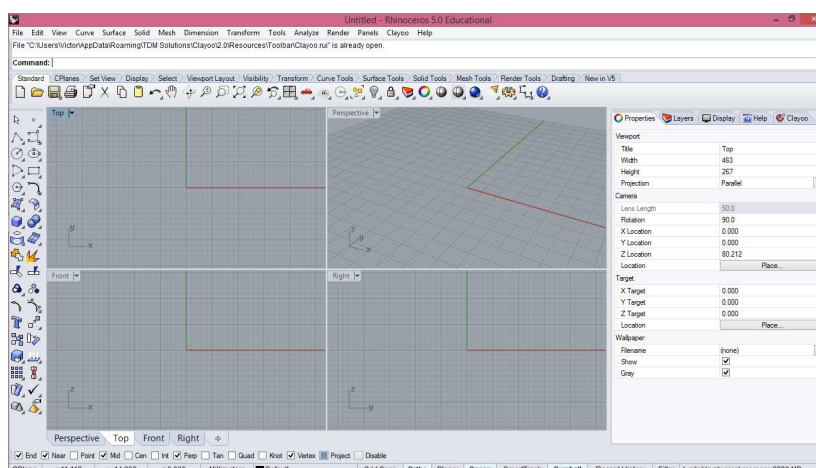
Armazenamento: 2.5"750 GB 5400 RPM / 24GB SSD Cache

Software

Os pacotes de software utilizados como plataforma do desenvolvimento dos algoritmos foram:

Rhinoceros 3D (versão: SR13 64-bit (5.13.60523.20140)):

Figura 103 – Interface Rhinoceros 3D



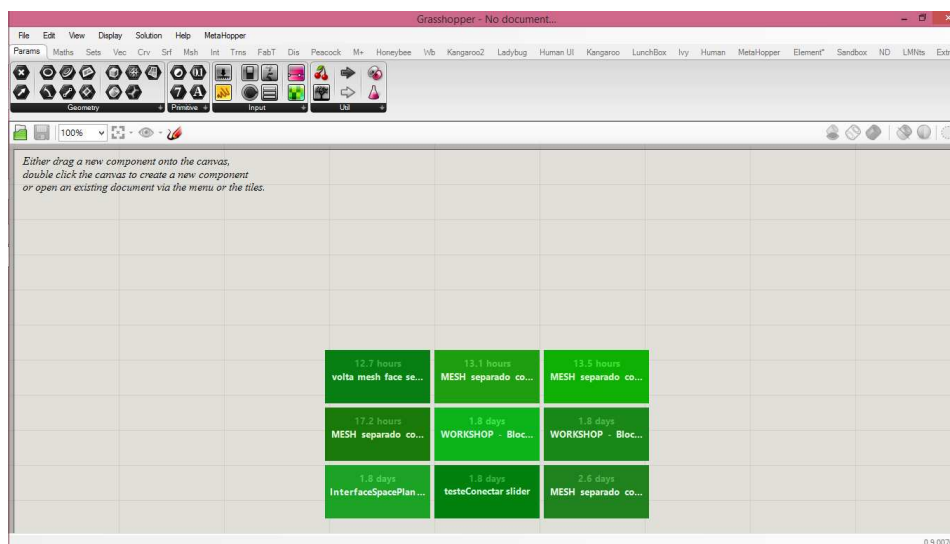
Fonte: Do autor

Software de modelagem 3D Nurbs desenvolvido por Robert McNeel & Associates.

Grasshopper (versão: 27 de Agosto de 2014 (0.9.0076))

Grasshopper é um editor gráfico de algoritmos integrado como *plug-in* ao Software de modelagem Rhinoceros 3D.

Figura 104 – Interface Grasshopper

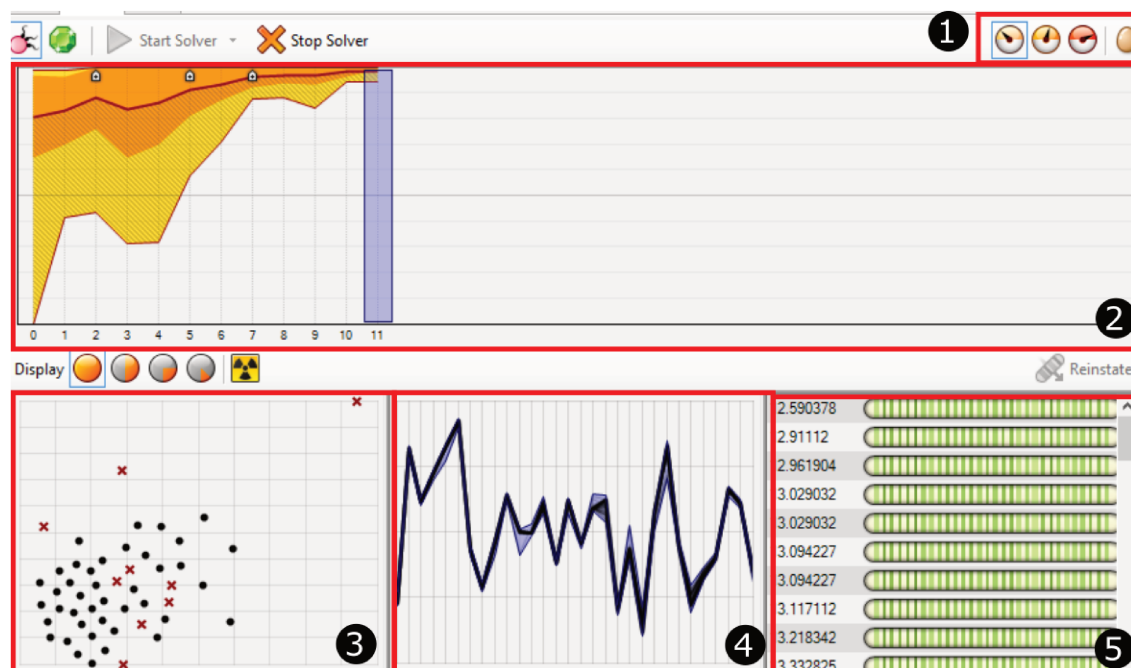


Fonte: Do autor

Galápagos

Galápagos é um *solver* evolutivo integrado à interface do Grasshopper. Na figura 105, a marcação com o número **1** são os modos de visualização dos fenótipos enquanto o *solver* está executando, se o botão da esquerda está ligado, isto significa que todos os indivíduos serão mostrados na tela do *Grasshopper* durante a seleção. Caso o botão do meio esteja ligado, somente os indivíduos com maior valor de aptidão de cada geração serão mostrados e, por fim, caso o botão da direita esteja ligado, não será mostrado nenhum fenótipo em nenhuma das gerações. Este modo serve para otimizar os cálculos poupando a placa gráfica do computador. A marcação número **2** é um gráfico que mostra a evolução dos indivíduos no decorrer das gerações. A parte inferior do gráfico representa o pior indivíduo da geração corrente, enquanto a parte superior do gráfico mostra o melhor indivíduo. A linha vermelha é a média dos valores de aptidão da geração, enquanto a parte alaranjada do gráfico é o desvio padrão. A marcação **3** mostra os indivíduos que estão sobrevivendo e os que estão morrendo no espaço solução de cada geração. O **4** é um gráfico que representa o valor de aptidão de cada solução por geração. Por fim, o **5** mostra o valor de aptidão por indivíduo, de todas as gerações, ordenado com os melhores indivíduos na parte superior da lista.

Figura 105 – Interface Galápagos



Fonte: Do autor

5.2 Métodos

5.2.1 Procedimentos para execução dos experimentos aplicados

Para a execução dos experimentos foram estabelecidos alguns procedimentos como protocolo, para que os dados pudessem ser analisados e comparados em uma etapa posterior. A escolha dos valores desses procedimentos se basearam em parâmetros padrões do galápagos, pois esses dados foram estabelecidos pelo desenvolvedor do *plug-in* porque tendem a ter resultados ponderados na maioria das simulações executadas.

Configurações do Galápagos

População: 50 indivíduos

Início da População: 100 indivíduos

Manter da população: 5% da população

Descendentes: 75% da população

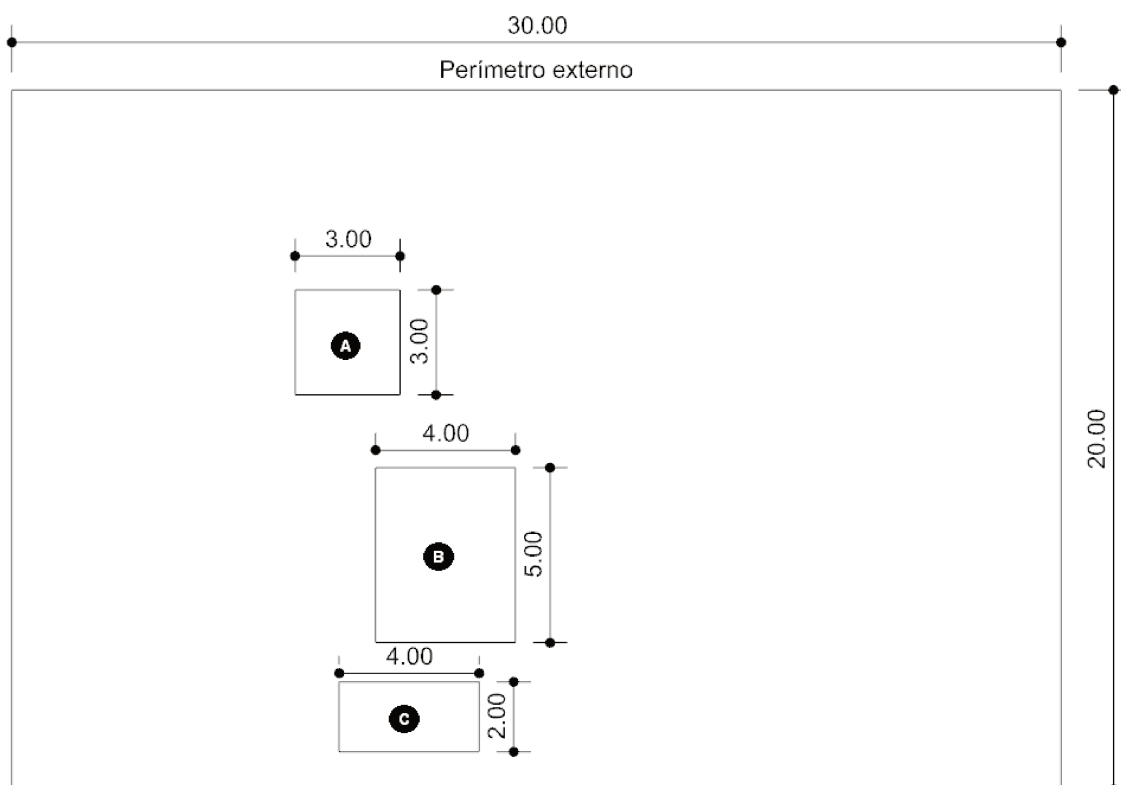
5.2.2 Método de implementação dos algoritmos

5.2.2.1 Agrupamento simples

A criação das regras dos experimentos foi desenvolvida através de um exercício de agrupamento simples, em que um perímetro maior continha os elementos de dimensão

menor através de certa disposição específica. Dado um perímetro externo e 3 elementos com área menor que o mesmo, foram estabelecidos e identificados os passos para que os elementos ficassem dispostos repetindo as regras pré-estabelecidas. O primeiro passo foi identificar quais seriam as restrições utilizadas para obter o resultado desejado (figura 106).

Figura 106 – Agrupamento simples



Fonte: Do autor

As regras estabelecidas foram :

01 - Os elementos não poderiam sair do perímetro externo.

02 - Os elementos não poderiam se sobrepor.

03 - Os elementos deveriam estar próximos a uma certa distância de um outro elemento de uma maneira pré-estabelecida.

Essas regras foram as bases do desenvolvimento dos experimentos posteriores.

Regra 01 - Não sair do perímetro externo

O primeiro passo para estabelecer essa regra foi buscar diminuir o espaço de busca do Galápagos. Dado que a localização dos elementos é feita através de coordenadas cartesianas XYZ, se o elementos não estivessem restritos dentro de um perímetro específico,

a busca de localização seria feita por todo o espaço tridimensional possível do *Rhinoce-ros 3D*, ou seja, por todas as possibilidades dentro da tolerância absoluta definida nas propriedades de unidade do *software* (figura 107).

Foi estabelecida, então, uma posição máxima para os elementos em relação ao perímetro externo que ficou estabelecida por:

World-XY máximo de E = (X-Ex,Y-Ey)

E = A,B,C.

Ex = dimensão x do elemento.

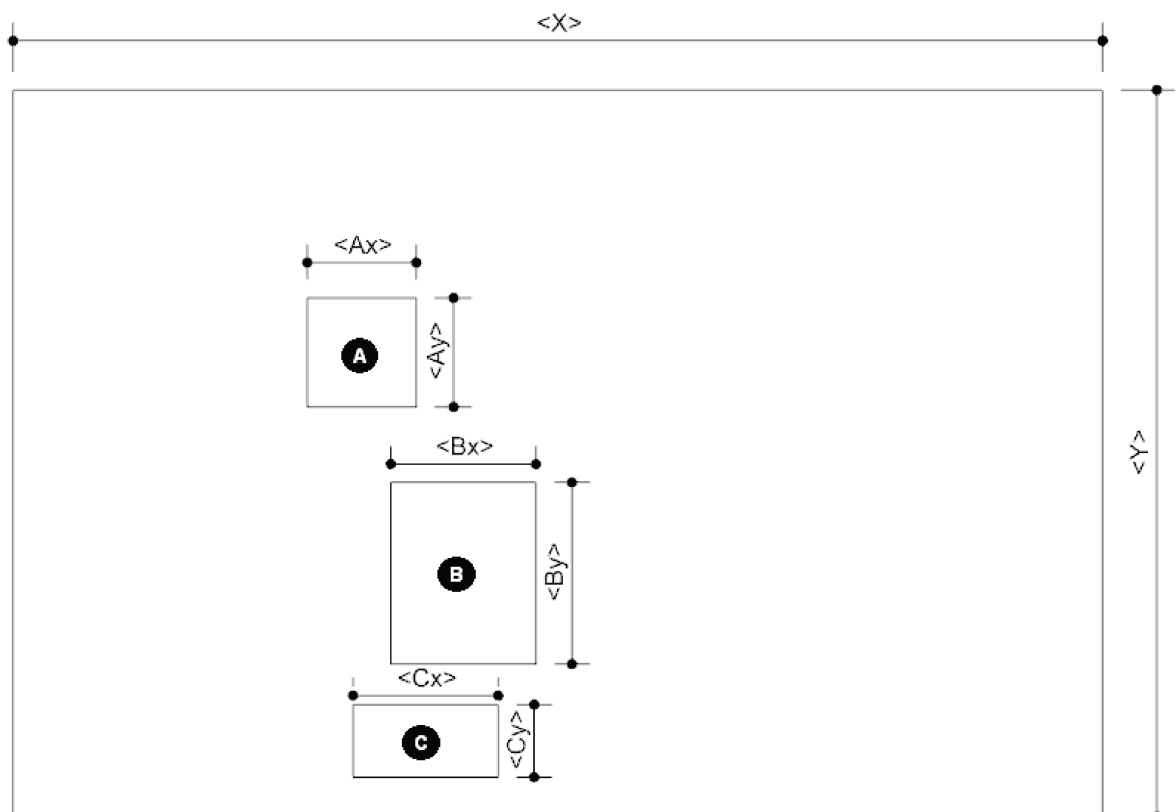
Ey = dimensão y do elemento.

X = dimensão X do perímetro externo.

Y = dimensão Y do perímetro externo.

World-XY = Dimensão absoluta de XY do espaço de tridimensional do *software*.

Figura 107 – Agrupamento simples com variáveis paramétricas



Fonte: Do autor

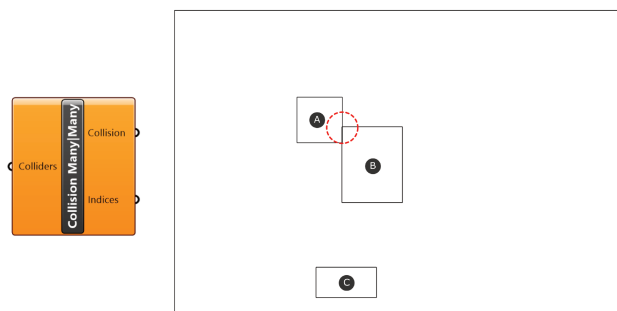
Regra 02 - Não haver colisão e sobreposição de elementos

A segunda regra foi estabelecida por dois parâmetros similares **colisão** e **so-**

breposição. A primeira regra verifica se existe colisão entre os elementos, enquanto a segunda verifica se há sobreposição de elementos. Apesar de similares, as regras atuam em situações diferentes, por exemplo, a primeira verifica se há choque entre os elementos, mas não garante a identificação quando um elemento menor está dentro de outro elemento, enquanto a segunda regra atua com auxiliar e garante que não haja a sobreposição dos elementos.

A primeira regra é implementada por um componente do *Grasshopper* que verifica colisão "*Colision many to many*" entre polígonos bidimensionais fechados (figura 108).

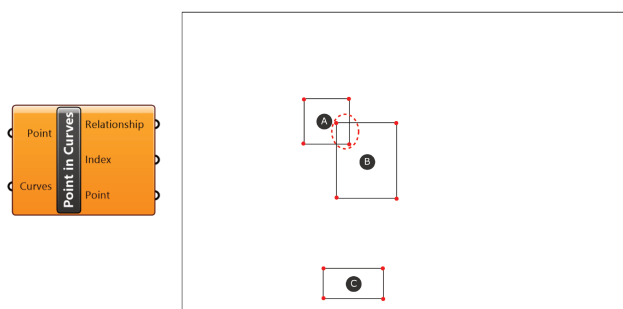
Figura 108 – Regra Colisão



Fonte: Do autor

A segunda regra é obtida por um componente do grasshopper que verifica a inclusão de pontos dentro de uma curva "*Points in curve*", a verificação é feita conferindo se há algum vértice do polígono que compõe os elementos dentro do perímetro (curva) de um segundo elemento (figura 109).

Figura 109 – Regra Sobreposição

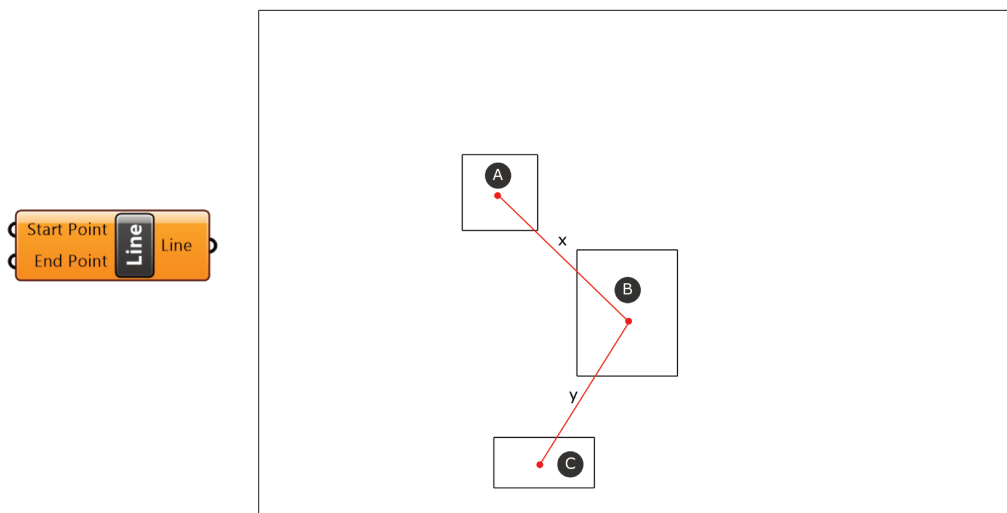


Fonte: Do autor

Regra 03 - Adjacência e distância entre os elementos

A terceira e última regra é feita através da construção de um grafo entre os elementos, em que as arestas definem as relações de adjacência e distância entre os elementos a partir do centro dos polígonos. O usuário define quais elementos são adjacentes no grupo e então, para cada relação, define as distâncias desejadas. O grafo é construído através do componente *Line* que traça uma linha entre dois pontos (*"Line 2 points"*) (figura 110). Na figura 111 é possível ver o resultado da aplicação dessas regras.

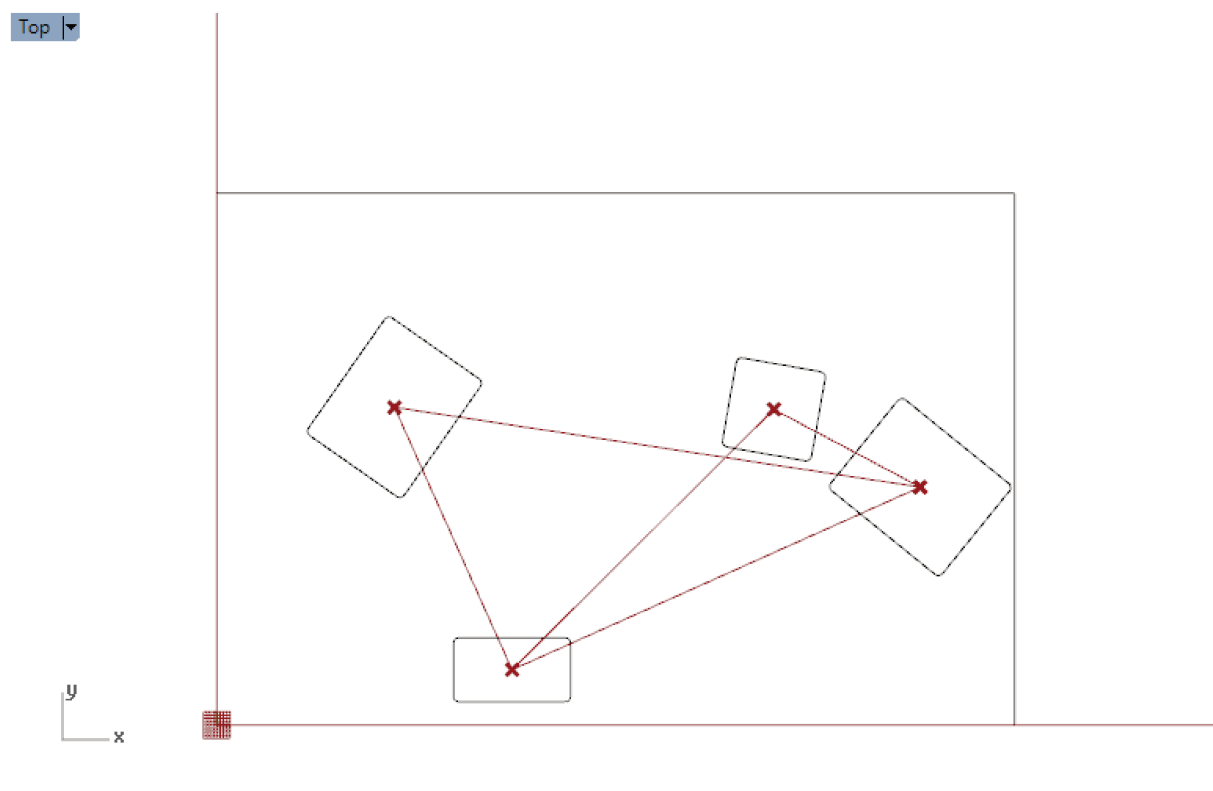
Figura 110 – Regra Adjacência



Fonte: Do autor

São estas 3 regras básicas que definiram este primeiro estudo e que foram a base do desenvolvimento dos experimentos.

Figura 111 – Primeiro Experimento



Fonte: Do autor

Parte V

Experimentos

6 Experimentos

Dada a complexidade da metodologia de *Space Planning* e de seus distintos métodos de representação e avaliação do espaço, uma das tarefas da implementação dos experimentos foi escolher os métodos que seriam aplicados. Por se tratar de uma pesquisa de mestrado, foge ao escopo deste trabalho implementar todos os métodos que foram descritos neste trabalho.

O modo de representação do espaço escolhido para o desenvolvimento dos experimentos foi o método de agrupamento (*"Blocking"*). No desenvolvimento de um projeto de arquitetura esse método pode ser bem versátil, pois delimita-se um perímetro externo e quais serão os elementos que estarão contidos dentro deste, o que possibilita trabalhar com distintas escalas e formas diversas, desde a disposição de mobiliários dentro de uma planta até a setorização de edifícios dentro de quadras em um projeto de desenho urbano.

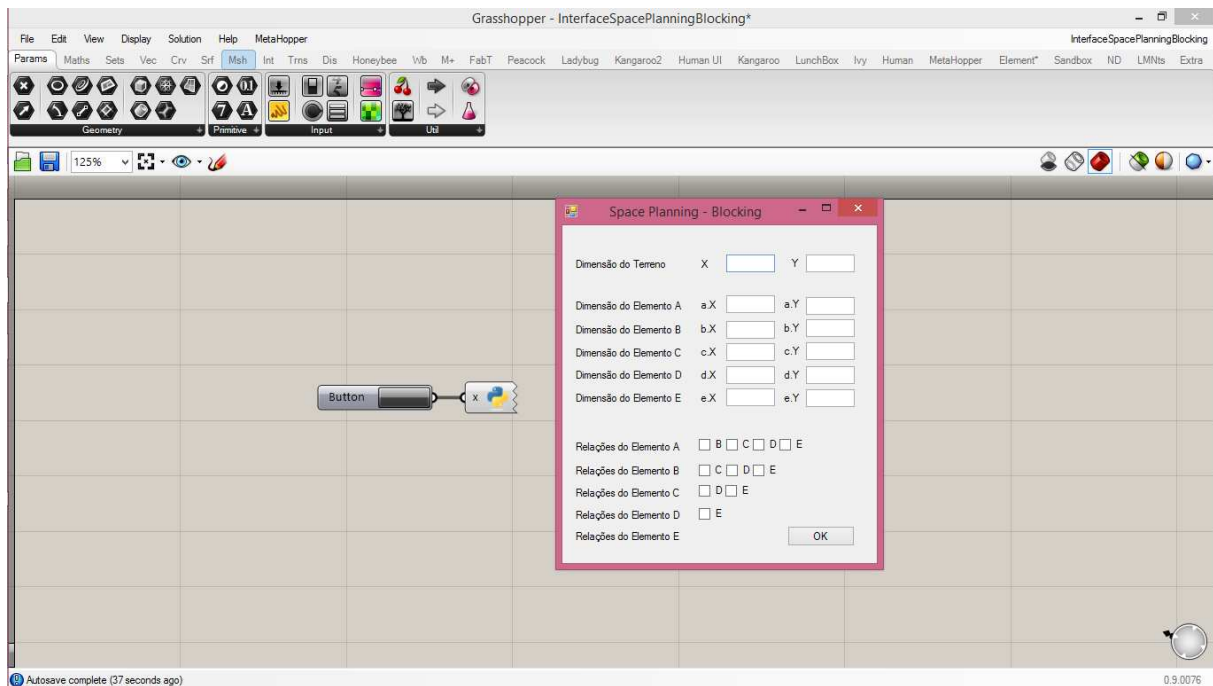
Os exercícios foram desenvolvidos com complexidade incremental e serão discutidas nos próximos tópicos.

6.0.1 Teste de Interface

Primeiro Teste de Interface - SharpDevelopment

Durante o desenvolvimento da criação das regras dos experimentos notou-se a necessidade da criação de uma interface que facilitasse a entrada de dados do usuário, tornando o algoritmo mais intuitivo para o usuário. O primeiro estudo de interface buscou através de linguagem *script* (*Python*) conectar a interface do *Grasshopper* com as *Windows Forms* do *Windows*, que são os métodos que criam as janelas dentro do Sistema *Windows*. Para isto foi usado um *software* de criação de interface chamado *SharpDevelopment*, que criava de maneira visual as janelas e devolvia como dado de saída um *script* com os dados necessários para a criação das janelas desenvolvidas (imagem 112).

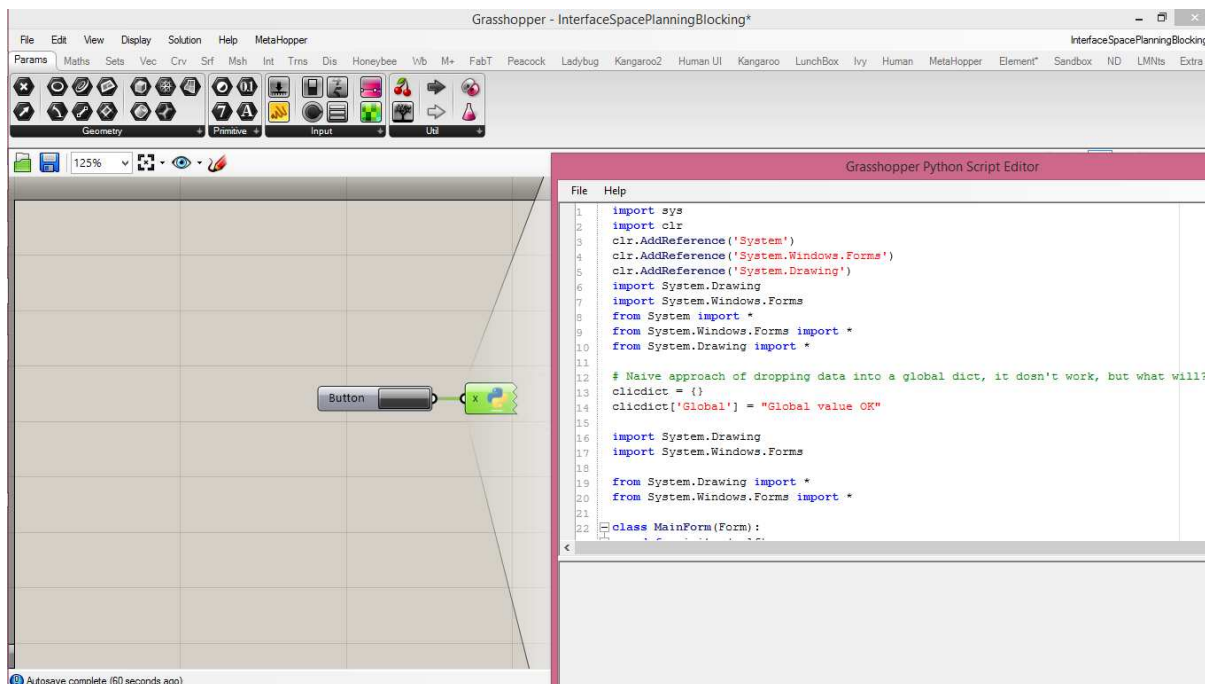
Figura 112 – Componente Python "chamando" janela



Fonte: Do autor

Desse modo, o componente de *script Python* do *Grasshopper* importava os módulos de *Windows forms*, juntamente com outros módulos necessários que podem ser vistos na figura 113.

Figura 113 – Módulos necessários para criação da janela

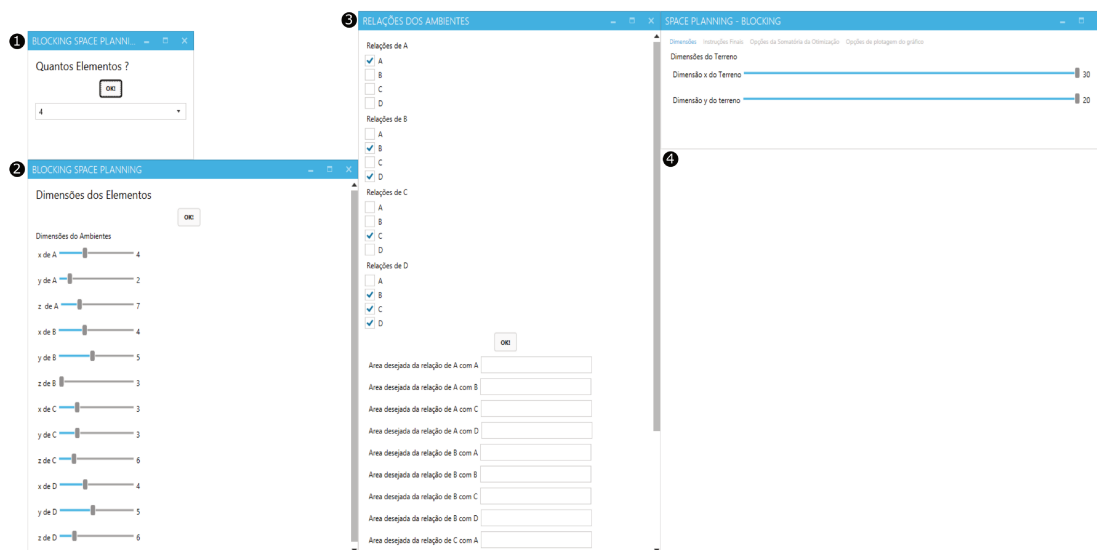


Fonte: Do autor

Assim foi possível, em um primeiro momento, criar uma interface mais intuitiva para o usuário, mas ainda faltava a resolução de outros quesitos como, por exemplo, tipo do dado de saída, conexão entre as janelas, entre outros. Esses tópicos foram solucionados no segundo teste.

Segundo Teste de Interface - *Human UI*

Enquanto estava sendo desenvolvida a interface no *SharpDevelopment*, foi lançado na fórum do *Grasshopper* um *plug-in* que faz a conexão entre *Windows Forms* e *Grasshopper*, exatamente com o mesmo objetivo que estava sendo desenvolvido para esta dissertação, mas com várias possibilidades a mais implementadas. Isto possibilitou um avanço no desenvolvimento da interface do algoritmo, além de economizar alguns passos em seu desenvolvimento. A partir desse momento a interface do algoritmo base passou a ser desenvolvida dentro do *plug-in* de maneira integrada ao ambiente do *Grasshopper* (imagem 114). Através deste *plug-in* foi possível solucionar os problemas anteriores de conexão entre janelas e o tipo de dado de saída.

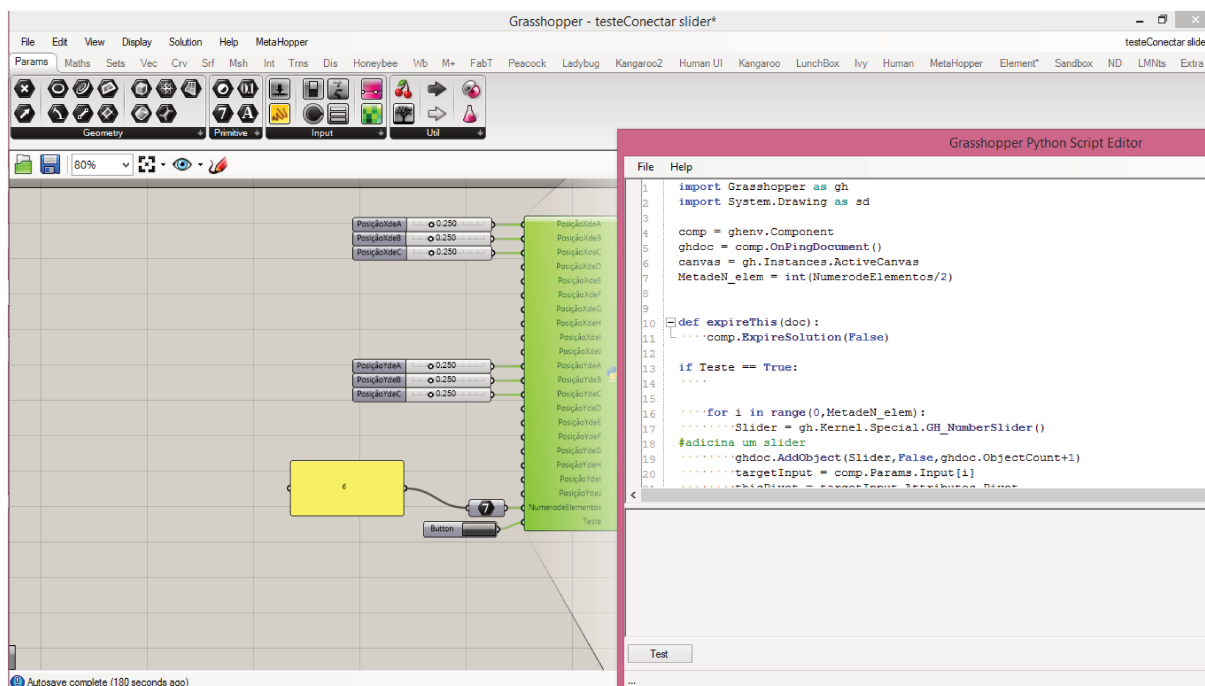
Figura 114 – Sequência de janelas do *plug-in Human UI*

Fonte: Do autor

6.0.2 Teste de Automatização de componentes

Primeiro Teste de Automatização de Componentes

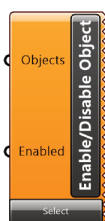
Com a criação da interface surgiram outras questões que deveriam ser solucionadas no algoritmo, pois após ser criada a interface, as janelas deveriam responder de maneira interativa aos dados de entrada que o usuário selecionava. O *Grasshopper* trabalha através de um fluxo de dados estático, onde não é possível através dos componentes padrão ligar ou desligar operações do fluxo de dados. Então, como tornar a interface interativa, respondendo aos dados de entrada que o usuário selecionava? Foi necessário em um primeiro momento buscar através de linguagem *script* os métodos que controlavam atributos dos componentes do *Grasshopper* como, por exemplo, posição, conexão, valores, entre outros. Foram feitos alguns testes bem sucedidos após a pesquisa dos métodos que controlavam os componentes do *Grasshopper* que podem ser vistos na figura 115. Neste exemplo foi automatizada a conexão dos *sliders* (que é um componente que gera valores numéricos ao deslizar uma barra) através de um componente *script* específico, com um valor desejado e em uma posição pré-definida do *canvas* (janela onde é desenvolvido o algoritmo do *Grasshopper*), mas enquanto essa etapa do trabalho estava sendo desenvolvida, foi lançado no fórum do *Grasshopper* um *plug-in* com o mesmo propósito, chamado *Metahopper*, e foi esse *plug-in* que foi utilizado na segunda etapa de desenvolvimento da automatização dos componentes.

Figura 115 – Automatização dos componentes do *Grasshopper* através de *script*

Fonte: Do autor

Segundo Teste de Automatização de Componentes - *Metahopper*

Com o *plug-in Metahopper* e um componente *Python* para o controle do fluxo de dados, foi possível controlar os elementos do algoritmo que deveriam estar ligados ou desligados, de acordo com as escolhas que o usuário realizava. O componente utilizado do *plug-in Metahopper* foi o "*Enable/Disable Objects*" (figura 116). Este liga e desliga componentes que foram previamente selecionados.

Figura 116 – Componente "*Enable/Disable Objects*" do *Metahopper*

Fonte: Do autor

6.1 Experimentos aplicados

Todas as etapas anteriores implementadas (definição das regras, desenvolvimento de interface e automatização do fluxo dos componentes) foram essenciais para o desenvolvimento dos exercícios aplicados. Foram desenvolvidos uma série de exercícios práticos que serviram como prova de conceito do método estudado.

A seguir são apresentados os exercícios desenvolvidos : implantação bidimensional, estudo de *layout*, implantação com torres tridimensionais, implantação com torres tridimensionais com otimização das visuais e implantação tridimensional com blocos modulares.

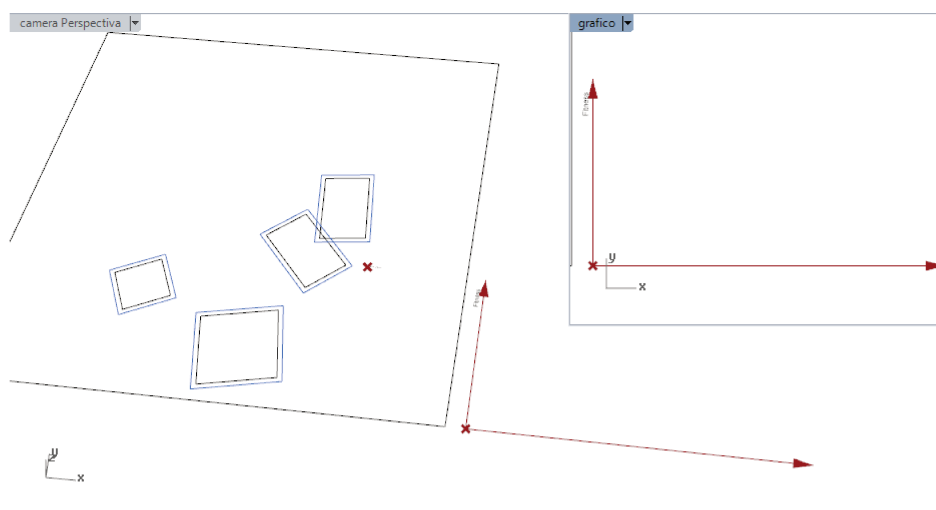
Todos os algoritmos desenvolvidos nos experimentos estão disponibilizados no CD-ROM que acompanha este volume.

6.1.1 Experimento 1- Implantação bidimensional

O experimento de implantação bidimensional foi desenvolvido em um terreno regular de 10000 m^2 , com elementos (retângulos) dispostos dentro desse perímetro. Cada elemento tinha a liberdade de rotação de 360 graus e respeitava as seguintes regras:

- 1- Não sair do terreno ;
- 2- Minimizar o somatório das distâncias dos elementos em relação a um ponto de atração ;
- 3- Minimizar a distância absoluta desejada pelo usuário nas relações de adjacência definidas através de um grafo plano.

Figura 117 – Terreno da implantação bidimensional



Fonte: Do autor

Os seguintes dados de entrada foram testados:

Quatro elementos definidos como: A, B, C e D (figura 118);

Dimensões dos elementos: A = 16x16 m, B= 10x17 m, C= 10x17 m e D= 10x10 m;

Relações de adjacência e distância desejada entre os elementos: A-B= 10 m, A-C= 8 m, B-C= 12 m, A-D= 8 m e C-D= 10 m .

Figura 118 – Janela de configuração dos elementos

BLOCKING SPACE PLANNING

Quantos Elementos terá a sua implantação ?

4

OK

NOME DOS ELEMENTOS

Qual o nome do elemento n 1 ? A

Qual o nome do elemento n 2 ? B

Qual o nome do elemento n 3 ? C

Qual o nome do elemento n 4 ? D

OK

DIMENSÕES DO AMBIENTES

OK

Dimensões do Ambientes

x de A 16

y de A 16

x de B 10

y de B 17

x de C 10

y de C 17

x de D 10

y de D 10

RELAÇÕES DOS ELEMENTOS

☒ A tem relação com B

☒ A tem relação com C

☒ B tem relação com C

☒ A tem relação com D

☐ B tem relação com D

☒ C tem relação com D

Qual a distância desejada da relação de A com B 10

Qual a distância desejada da relação de A com C 8

Qual a distância desejada da relação de B com C 12

Qual a distância desejada da relação de A com D 8

Qual a distância desejada da relação de C com D 10

OK

Fonte: Do autor

O experimento foi desenvolvido com o método de soma ponderada e os dados do algoritmo evolutivo (Galápagos) foram :

Fator de Colisão = 100;

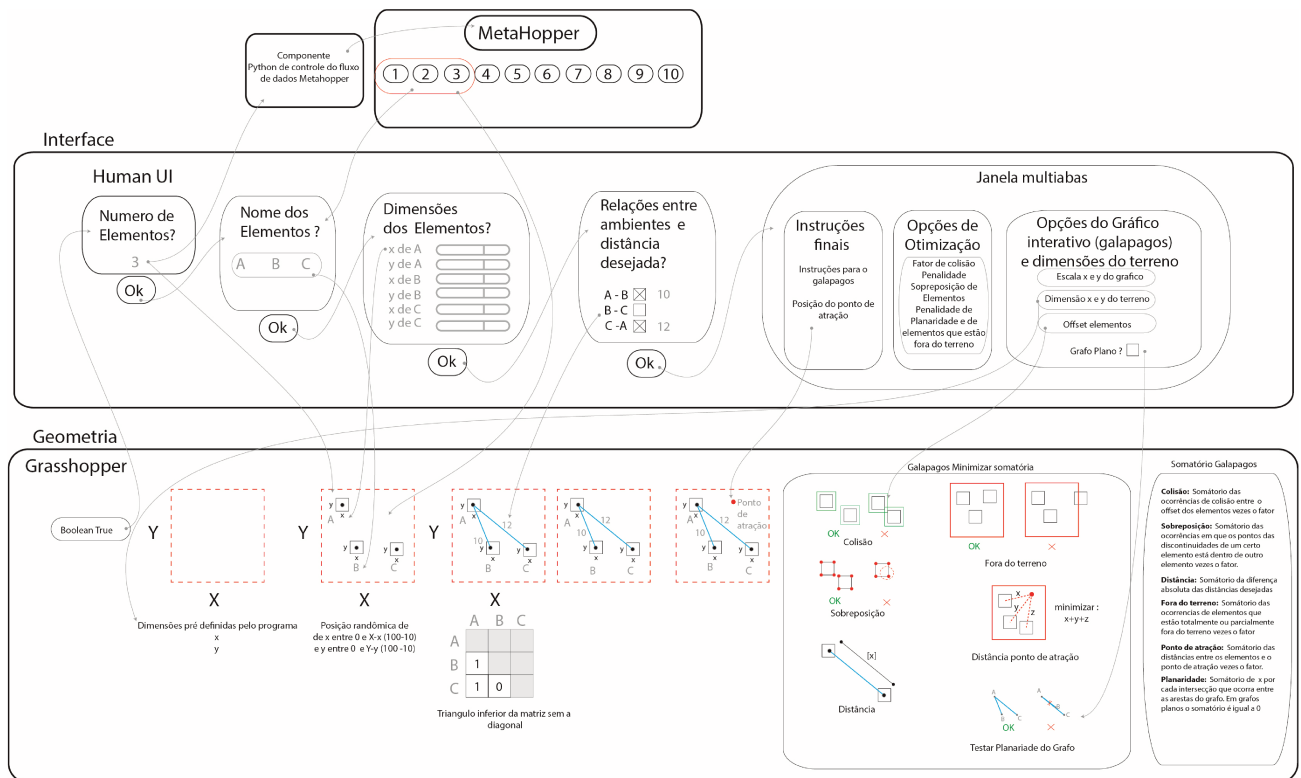
Penalidade Sobreposição de Elementos = 50;

Fator de Penalidade de Planaridade do Grafo = 30;

Fator de Penalidade de elemento fora do terreno = 100.

O algoritmo desenvolvido é descrito em forma de infográfico na Figura 119, em um exemplo que foram utilizados 3 elementos, e que ajuda a compreender o fluxo de dados do mesmo.

Figura 119 – Infográfico do fluxo de dados do algoritmo de implantação bidimensional

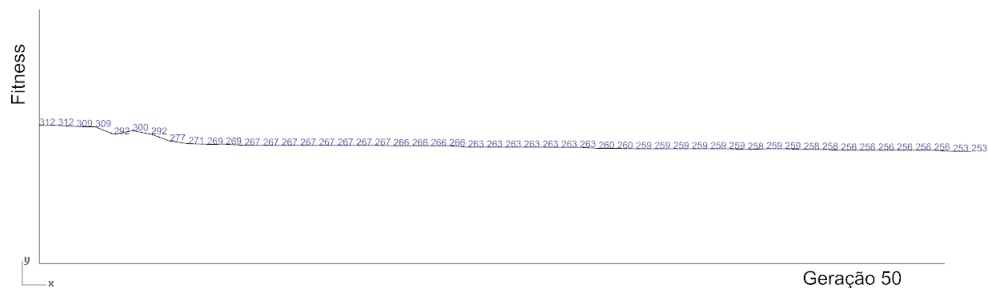


Fonte: Do autor

Resultados

O experimento atingiu 50 gerações em 27 minutos. O melhor resultado para a primeira geração foi o somatório de 312 pontos de aptidão, sendo que na geração 50 a mesma atingiu 253 pontos, obtendo uma melhora da solução em 59 pontos em relação à primeira geração, sendo que o somatório atingiu a convergência da solução na geração 9.

Figura 120 – Gráfico das 50 gerações



Fonte: Do autor

O resultado final foi a soma das distâncias das adjacências entre os ambientes e das distâncias do centro de cada elemento em relação a um ponto de atração. O algoritmo foi desenvolvido para minimizar a distância absoluta entre os elementos adjacentes buscando

Tabela 8 – Representação e valores da taxa de proximidade

Adjacências	Distâncias desejadas	Distância atingida	Desvio
A-B	10 m	26.6	16.6
A-C	8 m	35.3	27.3
B-C	12 m	13.4	1.4
A-D	8 m	20.1	12.1
C-D	10 m	19	9

Fonte: Do autor

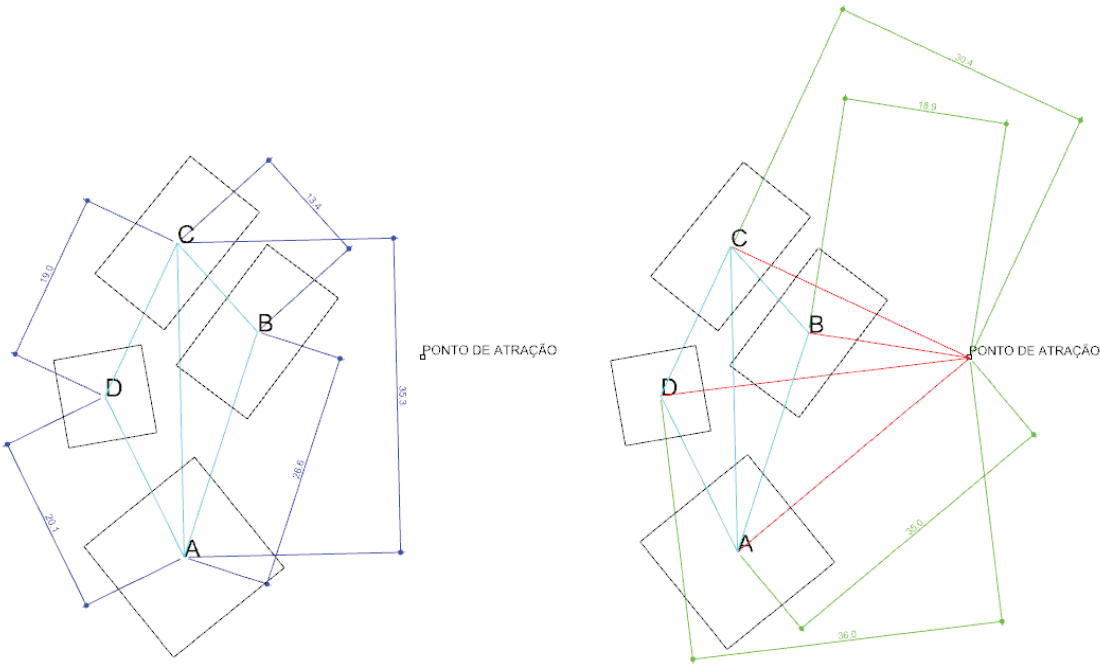
Tabela 9 – Distância entre os centros dos elementos e o ponto de atração

Elementos	Distância
A	36 m
B	18.9 m
C	30.4 m
D	35 m

Fonte: Do autor

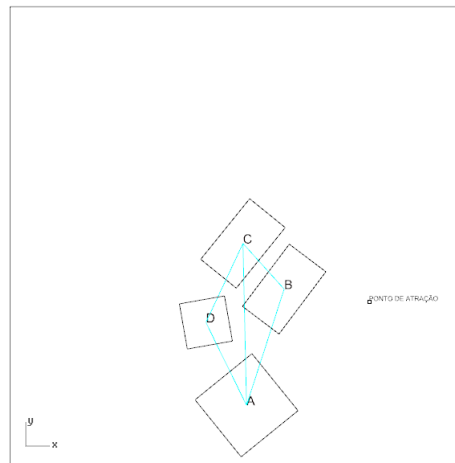
o número desejado pelo usuário, e caso o mesmo seja introduzido pelo usuário, o algoritmo não aplica penalidade ao somatório final. Caso não seja igual à distância desejada, a penalidade é igual à distância entre os elementos (tabela 8). O cálculo da penalidade do ponto de atração é igual ao somatório das distâncias (tabela 9) do centro de cada elemento em relação ao ponto de atração (figura 121).

Figura 121 – Distâncias entre elementos adjacentes e ponto de atração



Fonte: Do autor

Figura 122 – Resultado final do experimento



Fonte: Do autor

O resultado final foi satisfatório em relação à regra de "**1 - Não sair do terreno**", mas não foi bem sucedido com relação à regra "**2- Minimizar o somatório das distâncias dos elementos em relação a um ponto de atração**", pois a média dos valores ficou muito acima do valor desejado definido.

6.1.2 Experimento 2 - Estudo de *Layout*

O experimento de *layout* teve como base o algoritmo de implantação bidimensional, mas utilizou o retângulo base sem considerar a penalidade de colisão e sobreposição, sendo utilizado somente para geração do grafo plano que deu origem ao *layout* arquitetônico. Como o algoritmo de *layout* não considera as penalidades de colisão e sobreposição, foi necessária a criação de uma regra para a verificação do número de ambientes introduzidos como dados de entrada. As regras para a geração do *layout* arquitetônico foram:

- 1- Minimizar área desejada absoluta de cada ambiente;**
- 2- Minimizar irregularidade da planta;**
- 3- Contagem do número de ambientes;**

Os dados de entrada testados foram (figura 123) :

Quatro ambientes definidos como: A,B,C E D;

Relações de adjacência entre os ambientes: A-B, A-C, B-C, B-D e C-D;

Área desejada dos ambientes: A= 10, B= 15, C= 10, D= 8.

O experimento foi desenvolvido com o método de soma ponderada e os dados do algoritmo evolutivo (Galápagos) foram:

Fator de penalidade de planaridade do grafo = 30;

Penalidade de área desejada = 1;

Penalidade de menos ambientes = 500;

Penalidade por irregularidade da planta = 23.

Figura 123 – Janela de configuração dos elementos *layout*

The figure displays four screenshots of a software interface for layout configuration, arranged in a 2x2 grid. Each window has a blue title bar and standard window controls (minimize, maximize, close).

- Top Left: BLOCKING SPACE PLANNING**
Title: "Quantos Ambientes terá o seu layout ?"
Content: A text input field containing the number "3" and an "OK" button.
- Top Right: RELAÇÕES DOS AMBIENTES**
Content: A list of six checkboxes with labels: "A tem relação com B", "A tem relação com C", "B tem relação com C", "A tem relação com D", "B tem relação com D", and "C tem relação com D". The first five are checked, and the last one is unchecked. An "OK" button is at the bottom right.
- Bottom Left: NOME DOS AMBIENTES**
Title: "Qual o nome do ambiente n 1 ?"
Content: Four rows of input fields for naming environments 1 through 4, and four rows of input fields for their desired areas. The values entered are: Environment 1 (A, 10), Environment 2 (B, 15), Environment 3 (C, 10), and Environment 4 (D, 8). An "OK" button is at the top right.
- Bottom Right: SPACE PLANNING - BLOCKING**
Content: A series of sliders and checkboxes for optimization settings. The sliders are: "Fator de Penalidade de Planaridade do Grafo" (set to 30), "Penalidade de Área desejada" (set to 1), and "Penalidade de menos ambientes" (set to 500). There is a checkbox labeled "Minimizar irregularidade da planta?" which is currently unchecked.

Fonte: Do autor

O algoritmo de subdivisão do grafo que foi usado nesse experimento, foi um componente de *Grasshopper* desenvolvido por Nourian *et al.* (2013) baseado em algoritmos de experimentos anteriores realizados por Steadman (1983), Roth e Hashimshony (1988).

O algoritmo foi ilustrado em um infográfico, em exemplo que foram utilizados 3 ambientes, e que nos ajuda a compreender o fluxo de dados do mesmo (figura 6.1.2).

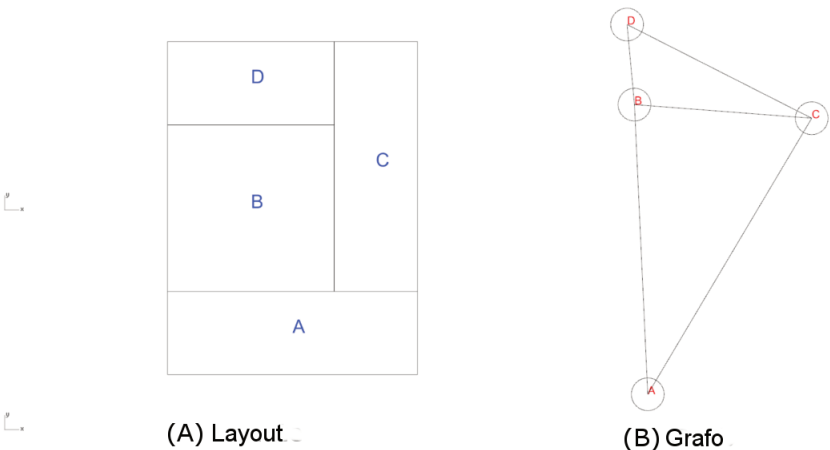
Tabela 10 – Ambientes e áreas desejadas e resultados

Ambientes	Áreas desejadas	Áreas atingidas	Desvios
A	10 m^2	10 m^2	0
B	15 m^2	13 m^2	2
C	10 m^2	10 m^2	0
D	8 m^2	7 m^2	1

Fonte: Do autor

veremos que a convergência para a solução final ocorreu na geração 7. Na tabela 10 vemos a comparação entre as áreas desejadas e as áreas obtidas.

Figura 125 – (a) Resultado do *layout* solução (b) Resultado do *layout* grafo



Fonte: Do autor

O experimento com *layouts* arquitetônicos comprovou ser viável com poucos ambientes a serem otimizados. Em um estudo mais criterioso, para além do escopo deste trabalho, seria necessário a aplicação de vários testes com distintas quantidades de ambientes para posteriormente se fazer uma análise comparativa dos resultados obtidos.

6.1.3 Experimentos 3 e 4 - Implantação com torres tridimensionais e implantação com torres tridimensionais com otimização das visuais

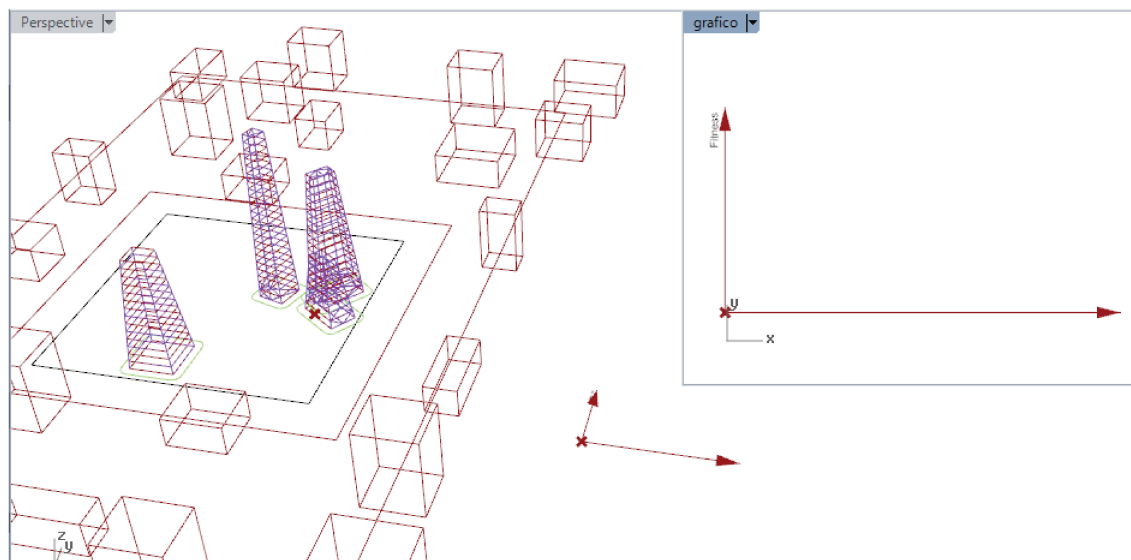
Os experimentos 3 e 4 foram uma continuidade do primeiro experimento de implementação bidimensional, sendo o segundo uma evolução do primeiro estudo. Os experi-

mentos foram desenvolvidos em um terreno regular de 100m x 100m. As torres foram dispostas dentro deste perímetro, sendo que, exatamente como no experimento 1, cada elemento tinha a liberdade de rotação de 360 graus.

Experimento 3

O experimento 3 - Implantação com torres tridimensionais - segue as mesmas regras da implementação bidimensional, mas com a diferença de que neste experimento é possível dar um valor de altura para as plantas-base. Foi introduzido um contexto de entorno que foi utilizado no experimento posterior.

Figura 126 – Terreno da implantação com torres tridimensionais



Fonte: Do autor

Os dados de entrada testados foram (figura 127):

Quatro elementos definidos como: A, B, C e D (figura 126);

Dimensões dos elementos : A= 16x16x42 m, B= 10x17x51 m, C= 10x17x25 m e D= 10x10x67 m;

Relações de adjacência e distancia desejada entre os elementos : A-B= 10 m, A-C= 5 m, B-C= 20 m, A-D= 10 m e C-D= 12 m .

Figura 127 – Dados de entrada implantação tridimensional

The figure displays four screenshots of the 'SPACE PLANNING - BLOCKING' software interface, showing the input parameters for a 3D layout optimization experiment.

Top Left: BLOCKING SPACE PLANNING
 Quantos Elementos?
 OK

Top Right: SPACE PLANNING - BLOCKING
 Dimensões Instruções Finais Opções da Somatória da Otimização Opções de plotagem do gráfico
☒ Grafo Pleno?
 Fator de Penalidade de Colisão: 15
 Penalidade Sobreposição de Elementos: 30
 Fator de Penalidade de Planaridade do Grafo: 30
 Fator de penalidade fora do terreno: 100

Bottom Left: BLOCKING SPACE PLANNING
 Dimensões dos Elementos
 Dimensões do Ambiente
 x de A: 16
 y de A: 16
 z de A: 42
 x de B: 10
 y de B: 17
 z de B: 51
 x de C: 10
 y de C: 17
 z de C: 25
 x de D: 10
 y de D: 10
 z de D: 67

Bottom Right: SPACE PLANNING - BLOCKING
 Dimensões Instruções Finais Opções da Somatória da Otimização Opções de plotagem do gráfico
 Dimensões do Terreno
 Dimensão x do Terreno: 100
 Dimensão y do terreno: 100
 (3D visualization of four blue rectangular blocks on a gray plane)

Far Right: RELAÇÕES DOS AMBIENTES

☒ A tem relação com B
☒ A tem relação com C
☒ B tem relação com C
☒ A tem relação com D
☐ B tem relação com D
☒ C tem relação com D
 Área desejada da relação de A com B: 10
 Área desejada da relação de A com C: 5
 Área desejada da relação de B com C: 20
 Área desejada da relação de A com D: 10
 Área desejada da relação de B com D: 15
 Área desejada da relação de C com D: 12

Fonte: Do autor

O experimento foi desenvolvido com o método de soma ponderada e os dados do algoritmo evolutivo (*Galápagos*) foram (figura 127):

Fator de Colisão = 15;

Penalidade Sobreposição de Elementos = 30;

Fator de Penalidade de elemento fora do terreno = 100.

Resultados do experimento 3

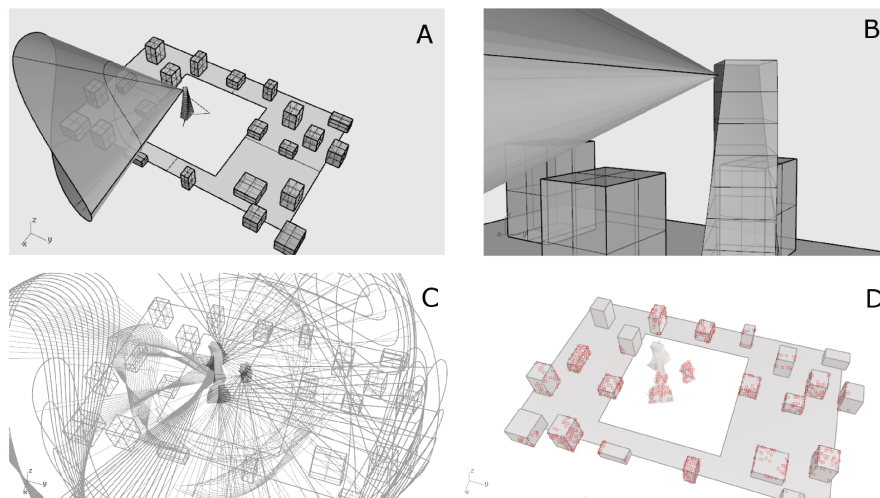
O experimento atingiu 50 gerações em 2 horas e 29 minutos. O melhor resultado para a primeira geração foi o somatório de 264 de aptidão, sendo que na geração 50 atingiu 168, obtendo uma melhora da solução de 96 pontos de aptidão em relação à primeira geração. O experimento mesmo com uma pontuação boa não apresentou bons resultados, pois os valores de distância desejados ficaram muitos distantes do resultado final (imagem 11). Esse experimento não calculou as distâncias do ponto de atração e nem a planaridade do grafo.

Experimento 4

O experimento 4 deu continuidade ao desenvolvimento do terceiro experimento. Esse experimento introduziu outras regras de composição que começaram a explorar a potencialidade de uso dos métodos de *space planning* com temáticas arquitetônicas. Trabalhou-se com o conceito de visuais arquitetônicas, definidas como:

- 1 - Rotação aleatória e interpolada de valores da base ao topo da torre, com o domínio variando entre 0 e 180 °;
- 2 - Cada andar de cada torre é subdividido em faces norte, sul, leste e oeste, sendo que, a partir do centro de cada face, é calculado o ângulo de visão de uma pessoa (120° no sentido horizontal e 60 ° no sentido vertical) através de uma malha poligonal (*mesh*);
- 3- As malhas poligonais dos ângulos de visão são interseccionadas com os edifícios do entorno e com a configuração atual das demais torres. Como resultado desta intersecção, é devolvida uma lista de pontos, que é utilizada para determinar a quantidade de obstrução visual de cada solução através da contagem de cada um dos pontos.

Figura 130 – Visuais e Pontos de Interseção - (A) Visual de uma torre e um andar (B) Detalhe da visual (C) Visuais do conjunto de torres de todos os andares (D) Pontos de interseção das visuais de todos os andares com as demais torres e entorno



Fonte: Do autor

Os dados de entrada testados foram:

Quatro elementos definidos como: A, B, C e D;

Dimensões dos elementos: A= 16x25x29 m, B= 10x17x54 m, C= 10x17x26 m e D= 10x10x47 m;

Relações de adjacência e distância desejada entre os elementos: A-B= 10 m, A-C= 12 m, A-D= 15 m, B-D= 18 m e C-D= 10 m.

Figura 131 – Janelas de dados de entrada

The figure displays four windows from the 'BLOCKING SPACE PLANNING' software:

- Quantos Elementos terá a implantação ?**: A window with a text input field containing the number '3' and an 'OK!' button.
- RELAÇÕES DOS AMBIENTES**: A window for defining relationships between environments A, B, C, and D. It includes checkboxes for 'A tem relação com B?', 'A tem relação com C?', 'B tem relação com C?', 'A tem relação com D?', 'B tem relação com D?', and 'C tem relação com D?'. Below these are input fields for 'Area desejada da relação de A com B' (10), 'Area desejada da relação de A com C' (15), 'Area desejada da relação de B com C' (20), 'Area desejada da relação de B com D' (12), and 'Area desejada da relação de C com D' (10). An 'OK!' button is at the top right.
- Dimensões dos Elementos**: A window showing sliders for the dimensions of environments A, B, C, and D. The sliders are labeled 'x de A', 'y de A', 'z de A', 'x de B', 'y de B', 'z de B', 'x de C', 'y de C', 'z de C', 'x de D', 'y de D', and 'z de D'. The values are: x de A: 16, y de A: 25, z de A: 30, x de B: 10, y de B: 17, z de B: 22, x de C: 10, y de C: 17, z de C: 49, x de D: 10, y de D: 26, z de D: 38. An 'OK!' button is at the top right.
- SPACE PLANNING - BLOCKING**: A window with tabs for 'Dimensões', 'Instruções Finais', 'Opções da Somatória da Otimização', and 'Opções de plotagem do gráfico'. The 'Dimensões' tab is active, showing sliders for 'Dimensão x do Terreno' (100) and 'Dimensão y do terreno' (100). Below the sliders is a 3D visualization of the space planning problem, showing several blue rectangular blocks on a gray base. The 'Opções da Somatória da Otimização' tab shows sliders for 'Fator de Penalidade de Colisão' (15), 'Penalidade Sobreposição de Elementos' (100), and 'Fator de penalidade fora do terreno' (100). The 'Opções de plotagem do gráfico' tab has a checkbox for 'Grafo Plano?' which is checked.

Fonte: Do autor

O experimento foi desenvolvido com o método de soma ponderada e os dados do algoritmo evolutivo (Galápagos) foram:

Fator de Colisão = 15;

Penalidade Sobreposição de Elementos = 100;

Fator de Penalidade de elemento fora do terreno = 100;

Grafo Plano;

Resultados do experimento 4.

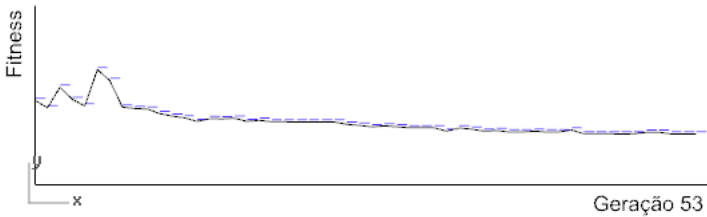
O experimento 4 atingiu 50 gerações em 1 hora. O melhor resultado para a primeira geração foi o somatório de 496 de aptidão, sendo que na geração 50 atingiu 410, obtendo uma melhora da solução de em 86 pontos de aptidão em relação à primeira geração.

Tabela 12 – Representação e valores da taxa de proximidade

Adjacências	Distâncias desejadas	Distâncias atingidas	Desvios
A-B	10 m	38 m	28 m
A-C	15 m	30 m	15 m
B-C	20 m	30 m	10 m
B-D	12 m	41 m	29 m
C-D	10 m	20 m	10 m

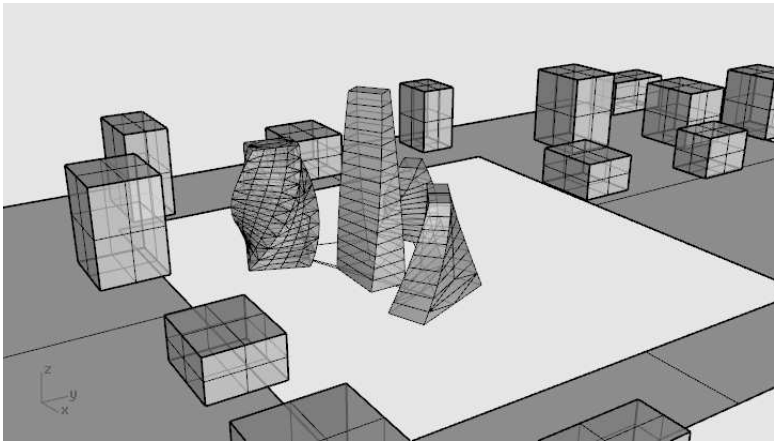
Fonte: Do autor

Figura 132 – Gráfico da implantação tridimensional com visuais



Fonte: Do autor

Figura 133 – Resultado final da implantação tridimensional com visuais



Fonte: Do autor

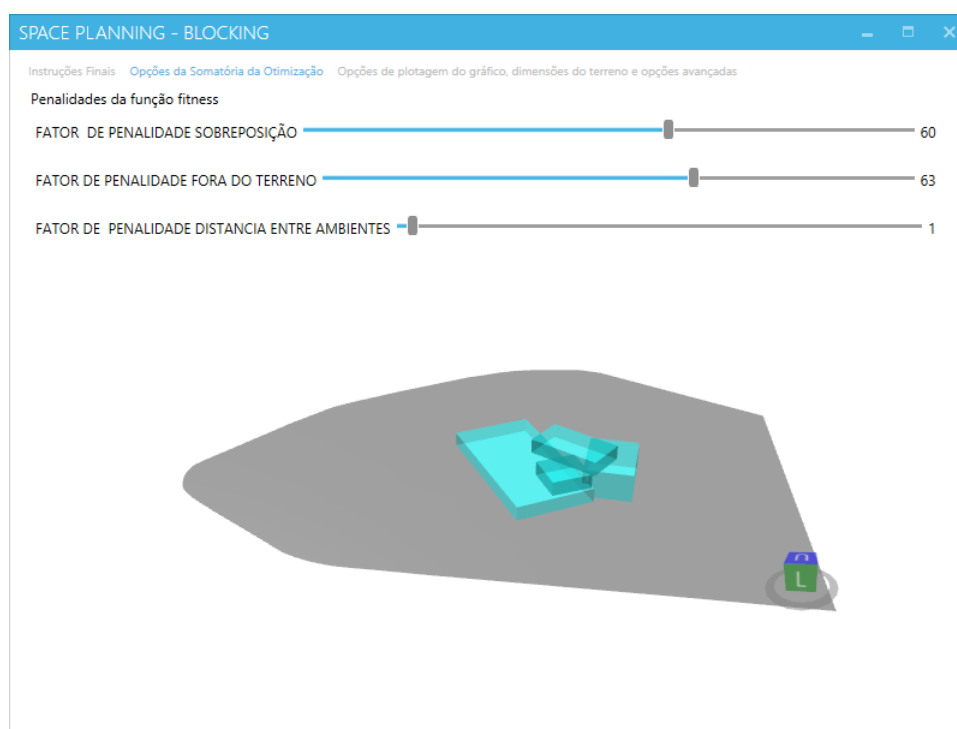
6.1.4 Experimento 5 - Implantação tridimensional com blocos modulares

O experimento 5 foi desenvolvido a partir do primeiro experimento, em que foi adicionada uma terceira dimensão aos elementos, criando blocos modulares em altura. Além disso, o terreno, que nos experimentos anteriores era regular, neste experimento passou a ser irregular, baseado em um polígono fechado, como geralmente os terrenos são na cidade real. O mesmo terreno utilizado aqui como teste foi utilizado posteriormente no Workshop descrito no próximo capítulo.

O experimento 5 foi desenvolvido em um terreno irregular (figura 134), em que blocos modulares (em altura) foram dispostos dentro desse perímetro, sendo definidos previamente os pavimentos em que cada elemento estaria. Além disto, cada elemento tinha uma liberdade de rotação de 360 graus, e respeitava às seguintes regras:

- 1- Não sair do terreno ;
- 2- Minimizar o somatório das distâncias dos elementos em relação a um ponto de atração ;
- 3- Minimizar a distância absoluta desejada pelo usuário nas relações de adjacência definidas pelo usuário através de um grafo plano;
- 4- Caso o elemento não estivesse no pavimento térreo, ao menos 2 vértices da superfície base do elemento deveriam estar dispostos acima da superfície superior de algum outro elemento; caso contrario, receberia uma penalidade por isto.

Figura 134 – Implantação inicial dos módulos e penalidades



Fonte: Do autor

Os dados de entrada testados foram (figura 134) :

O módulo de pé direito definido foi: 3.6 m

Dimensões dos elementos: A = 20x38x3.6 m, B= 12x16x7.2 m, C= 10x17x25 m e D= 10x10x3.6 m

Relações de adjacência e distância desejada entre os elementos: A-B= 10 m, A-C= 10 m, A-D= 15 m, B-D= 8 m e C-D= 10 m

O experimento foi desenvolvido com o método de soma ponderada e os dados do algoritmo evolutivo (galápagos) foram figura 135 :

Fator de penalidade de planaridade do grafo = 60;

Penalidade fora do terreno = 63;

Penalidade distância entre ambientes = 1.

Figura 135 – Dados de entrada implantação tridimensional com blocos modulares

BLOCKING SPACE PLANNING

Quantos Elementos terá a sua implantação ?
 Qual o módulo de pé direito adotado? 3,6
 OK

4

NOME DOS ELEMENTOS

Qual o nome do elemento n 1 ? A
 Qual o nome do elemento n 2 ? B
 Qual o nome do elemento n 3 ? C
 Qual o nome do elemento n 4 ? D
 Qual o nível do elemento n 1 ? Térreo
 Qual o nível do elemento n 2 ? Térreo
 Qual o nível do elemento n 3 ? Primeiro
 Qual o nível do elemento n 4 ? Segundo

RELACIONES DOS ELEMENTOS

☒ A tem relação com B
☒ A tem relação com C
☐ B tem relação com C
☒ A tem relação com D
☒ B tem relação com D
☒ C tem relação com D

Qual a distância desejada da relação de A com B 10
 Qual a distância desejada da relação de A com C 10
 Qual a distância desejada da relação de A com D 15
 Qual a distância desejada da relação de B com D 8
 Qual a distância desejada da relação de C com D 10
 OK

CONTROL WINDOW

OK

Quais são as dimensões dos elementos

x de A 20
 y de B 38
 módulos em z de A 1
 x de B 12
 y de B 16
 módulos em z de B 2
 x de C 10
 y de C 10
 módulos em z de C 1
 x de D 18
 y de D 10
 módulos em z de D 1

SPACE PLANNING - BLOCKING

Interações Forças | Capilares da Somatória da Orientação | Capilares de planagem do gráfico | dimensões do terreno e capilares associadas

Penalidades da função fitness

FATOR DE PENALIDADE SOBREPOSIÇÃO 60
 FATOR DE PENALIDADE FORA DO TERRENO 63
 FATOR DE PENALIDADE DISTANCIA ENTRE AMBIENTES 1

Fonte: Do autor

Resultados

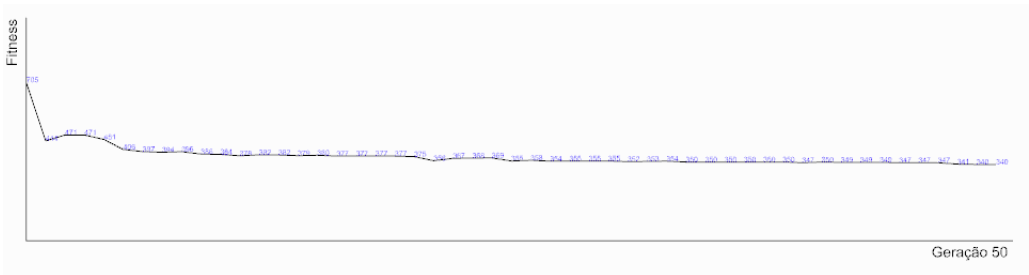
O experimento 5 atingiu 50 gerações em 23 minutos. O melhor resultado para a primeira geração foi o somatório de 750 pontos de aptidão, sendo que na geração 50 atingiu 340 pontos, obtendo uma melhora da solução de em 410 pontos em relação à primeira geração. O experimento, mesmo com uma pontuação boa, apresentou resultados medianos em 50 gerações, pois os valores de distância desejados ficaram distantes do resultado final (imagem 13).

Tabela 13 – Representação e valores da taxa de proximidade

Adjacências	Distâncias desejadas	Distâncias atingidas	Desvios
A-B	10 m	17 m	7 m
A-C	10 m	8 m	2 m
A-D	15 m	9 m	6 m
B-D	8 m	10 m	2 m
C-D	10 m	14 m	4 m

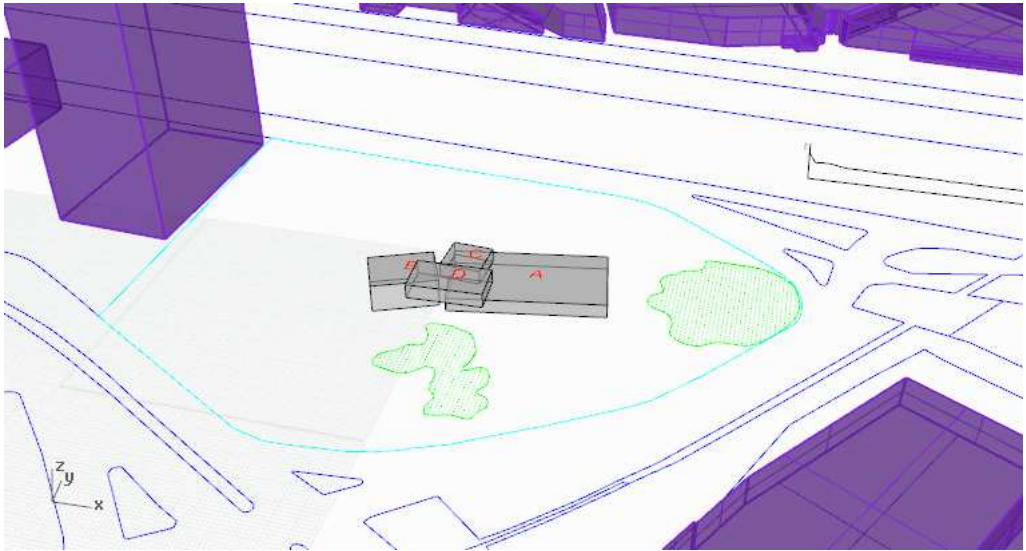
Fonte: Do autor

Figura 136 – Gráfico de implantação tridimensional com blocos modulares



Fonte: Do autor

Figura 137 – Resultado final da implantação tridimensional com blocos modulares



Fonte: Do autor

6.2 Considerações finais sobre os experimentos

Os experimentos tiveram como prioridade fazer testes das soluções dentro de um protocolo estabelecido. Os testes foram promissores nos estudos de casos realizados, embora ainda com baixa escalabilidade em relação ao número de elementos e variáveis testadas. Em trabalhos futuros poderão ser incluídas avaliações quantitativas e estatísticas dos métodos empregados. Para uma aplicação comercial seriam necessários outros testes, além de comparações entre os resultados, o que foge do escopo deste trabalho. Este trabalho buscou fazer uma primeira aproximação da temática, através da escolha de um método específico, com testes que serviram como primeiros experimentos para uma etapa que viria posteriormente, que foi o teste da aplicação através de um exercício prático, realizado através de um Workshop.

Parte VI

Workshop

7 Workshop

Esta etapa do trabalho teve como objetivo fazer uma análise qualitativa do método através de um exercício prático (Workshop), em que foram introduzidos os conceitos básicos de *space planning* juntamente com uma problemática arquitetônica de projeto. O Workshop foi realizado no dia 20/05/2016 na UNICAMP, LAPAC, e foi transmitido online via *google hangout* para participantes não presenciais. O registro editado pode ser acessado no *link* do youtube <<https://www.youtube.com/watch?v=MJv9BeVVb2w>>. As sessões a seguir descrevem o passo a passo deste Workshop, do planejamento à discussão dos resultados.

7.1 Planejamento do Workshop

O Workshop de *Space Plannning* teve como principal objetivo testar alguns dos algoritmos implementados através de um exercício de projeto, simulando em uma pequena escala problemáticas com as quais arquitetos se deparam no dia-a-dia de seus trabalhos. Para isso, foi pensada uma aplicação prática que consistia no desenvolvimento de uma implantação em um terreno existente da cidade de São Paulo, localizado próximo à estação intermodal da Barra Funda (figura 138). Foi feita uma enquete com possíveis participantes pré-selecionados para definir a melhor data para o Workshop, que ficou definida para o dia 20/05/2016, manhã e tarde. Após uma pré-seleção de participantes, foi feita uma chamada publica através das redes sociais para captar mais interessados e, dada a quantidade de interessados no tópico, foi definido que o Workshop seria transmitido e ministrado também via *Google Hangout*, permitindo que pesquisadores de outras regiões de São Paulo e do Brasil pudessem participar e contribuir com o trabalho.

O Workshop foi destinado a pessoas com nível intermediário/avançado de *Grasshopper*, pois não foram ministradas aulas sobre esse *software*. Foram selecionados 36 participantes, que participariam presencialmente e online, incluindo alunos da UNICAMP, pesquisadores e alunos de outras universidades de São Paulo e do Brasil. O material didático foi divulgado previamente na internet, com bibliografia básica e complementar e o pacote de *software* necessário, sendo que os algoritmos, planejamento do exercício e cronograma do curso foram inseridos posteriormente. Esses documentos encontram-se inseridos neste trabalho como Apêndices.

O Workshop foi dividido em duas partes. A primeira parte foi dedicada à instalação e explicação da funcionalidade dos *plug-ins*. Em seguida, foi feita uma apresentação introduzindo os principais conceitos de *space planning*. A segunda parte consistiu na realização de um exercício de *space planning* em grupos e na apresentação dos resultados

de cada um deles. A proposta do exercício foi a criação de um estudo de plano de massas para um complexo multifuncional na Barra Funda (figura 138), localizado na esquina da Av. Matarazzo com o Viaduto Antártica.

Figura 138 – Terreno do exercício proposto



Fonte: Do autor

O estudo deveria considerar a legislação urbana do local, bem como o programa de necessidades dado aos alunos. O principal objetivo dos grupos era utilizar os algoritmos de *space planning* fornecidos e incrementá-los com outras estratégias de projeto otimizados através do Galápagos.

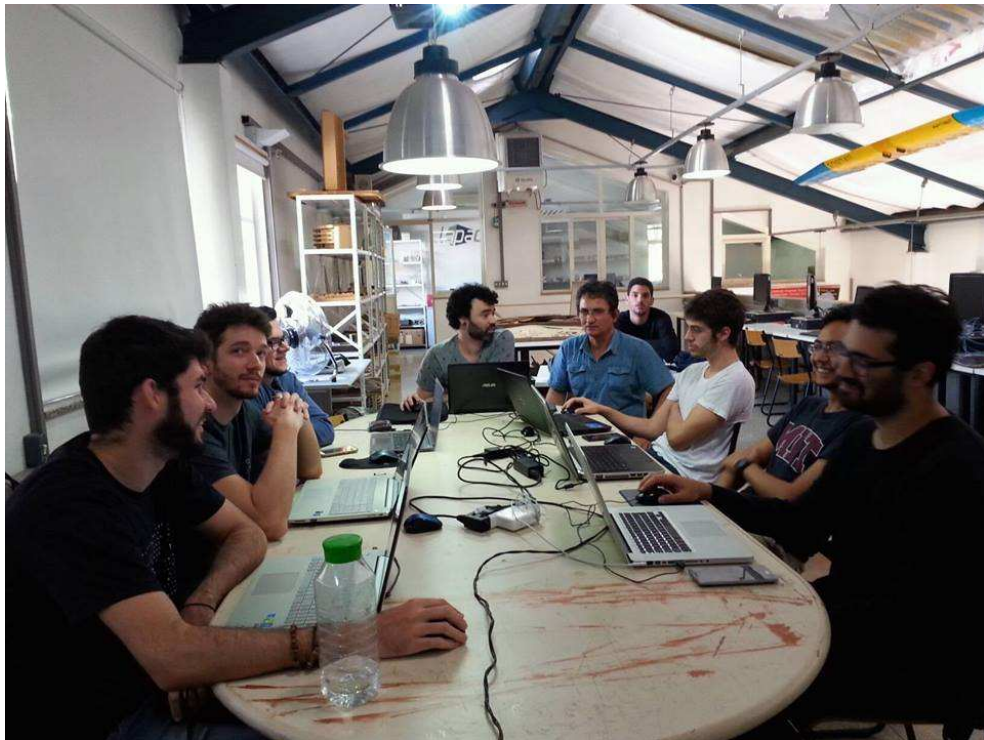
Os grupos deveriam levar em consideração as condicionantes locais para desenvolver a proposta e otimizar o edifício de acordo com as mesmas, por exemplo, visuais, áreas sombreadas, árvores do terreno, edifícios do entorno, acessos, etc. O programa de necessidades do exercício pode ser encontrado no apêndice deste trabalho.

7.2 Workshop

O Workshop contou com 8 alunos presenciais e 12 online. Na primeira parte do workshop foram apresentados os principais métodos do *Space Planning*, bem como conceitos sobre algoritmos evolutivos.

Na segunda parte, os participantes foram divididos em 5 grupos, 2 presenciais e 3 grupos online (figura 139). Todos eles foram auxiliados no desenvolvimento do exercício. Notou-se que os grupos que trabalharam presencialmente tiveram mais facilidade em desenvolver o exercício do que alguns grupos que trabalharam a distância. Isto se deu em consequência da plataforma utilizada para comunicação e pela lentidão da rede do local onde foi realizado workshop, problemas que são relativamente comuns em uma comunicação via internet.

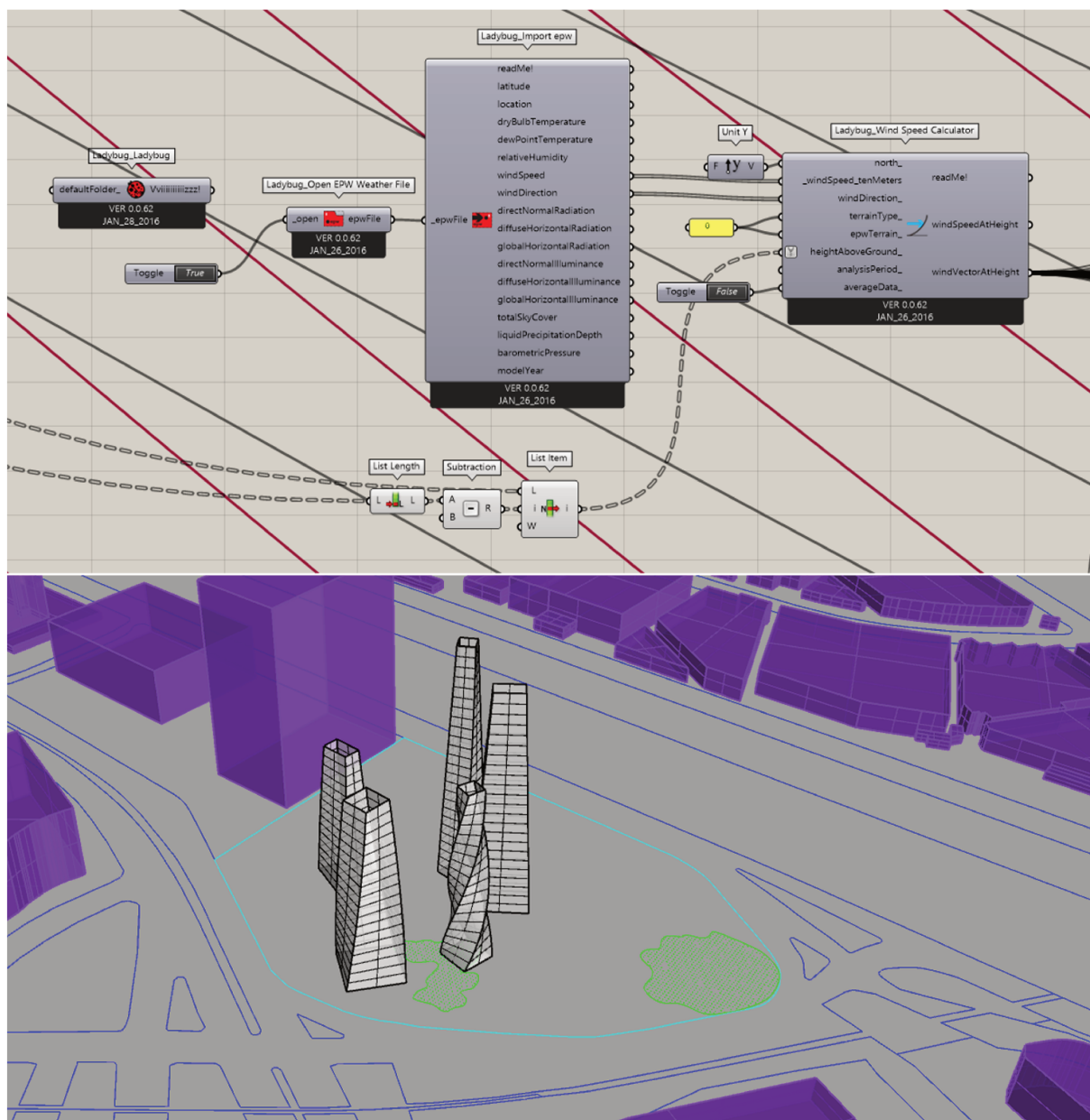
Figura 139 – Imagem do Workshop



Fonte: Do autor

De maneira geral, o Workshop aconteceu de uma maneira satisfatória, atendendo às expectativas e até as surpreendendo em alguns momentos, pelos resultados obtidos pelos participantes. Estes resultados serão discutidos a seguir na próxima seção.

Figura 141 – Algoritmo de simulação dos ventos e geometria produzida pelo grupo 1 - Felipe Tricoli, Gabriel Ribeiro, Rômulo Avi Oliveira e Vinicius Mizobuti



Fonte: Do autor

Grupo 2

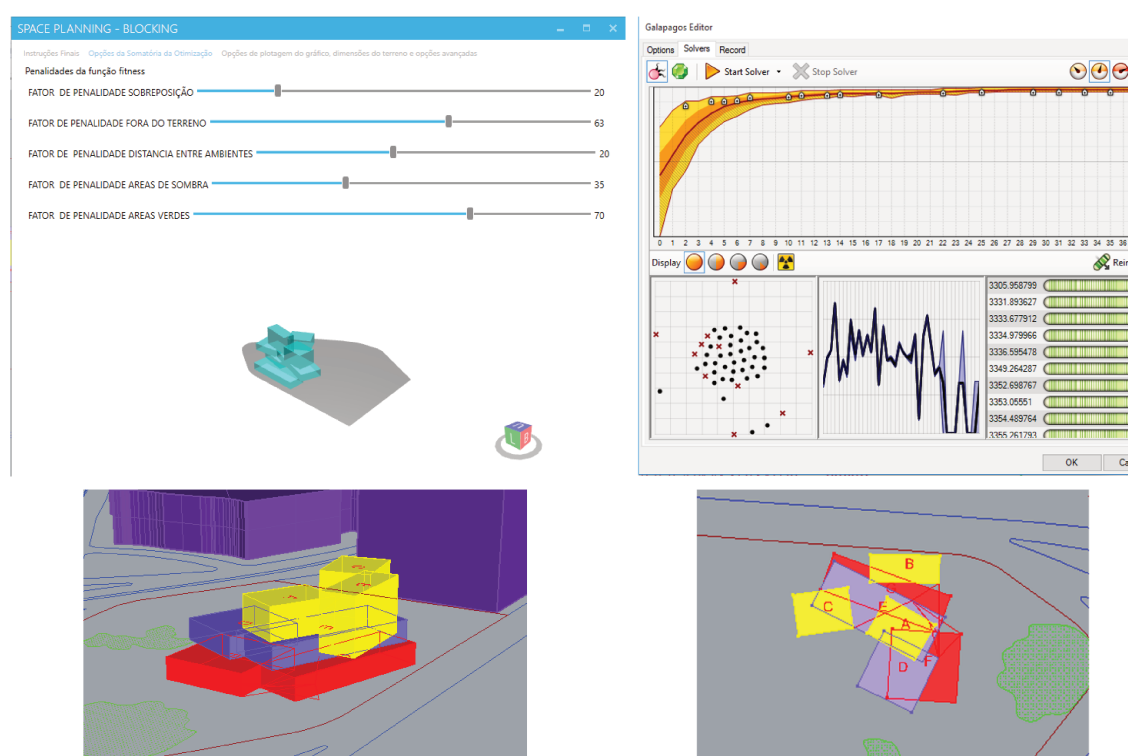
Participantes: Filipe Campos, Guilherme Giantini, Ronaldo A. Marques Pinto e Verley Henry Coco Jr.

O grupo 2 também participou presencialmente do workshop. Seus membros buscaram trabalhar com as seguintes temáticas: sombreamento e áreas verdes. O grupo desenvolveu alguns trechos de código em *Grasshopper*, que foram somados à base da implantação tridimensional com blocos modulares. A proposta do grupo foi trabalhar com a restrição de sobreposição das áreas verdes e com os pontos pouco iluminados devido do

entorno. O algoritmo desenvolvido pelo grupo definia usos de acordo com o programa de necessidade e atribuía uma cor para cada uso: amarelo para os blocos residenciais, roxo para os blocos de escritórios e vermelho para os blocos comerciais.

O grupo delimitou a área de sombreamento do entorno, sendo que se os blocos da solução invadissem esta área, seriam penalizados. Outra estratégia adotada pela equipe foi analisar distintas configurações de pesos das penalidades, permitindo uma melhor aproximação ao resultado desejado.

Figura 142 – Panorama geral do experimento do grupo 2 - Filipe Campos, Guilherme Giantini, Ronaldo A. Marques Pinto e Verley Henry Coco Jr



Fonte: Do autor

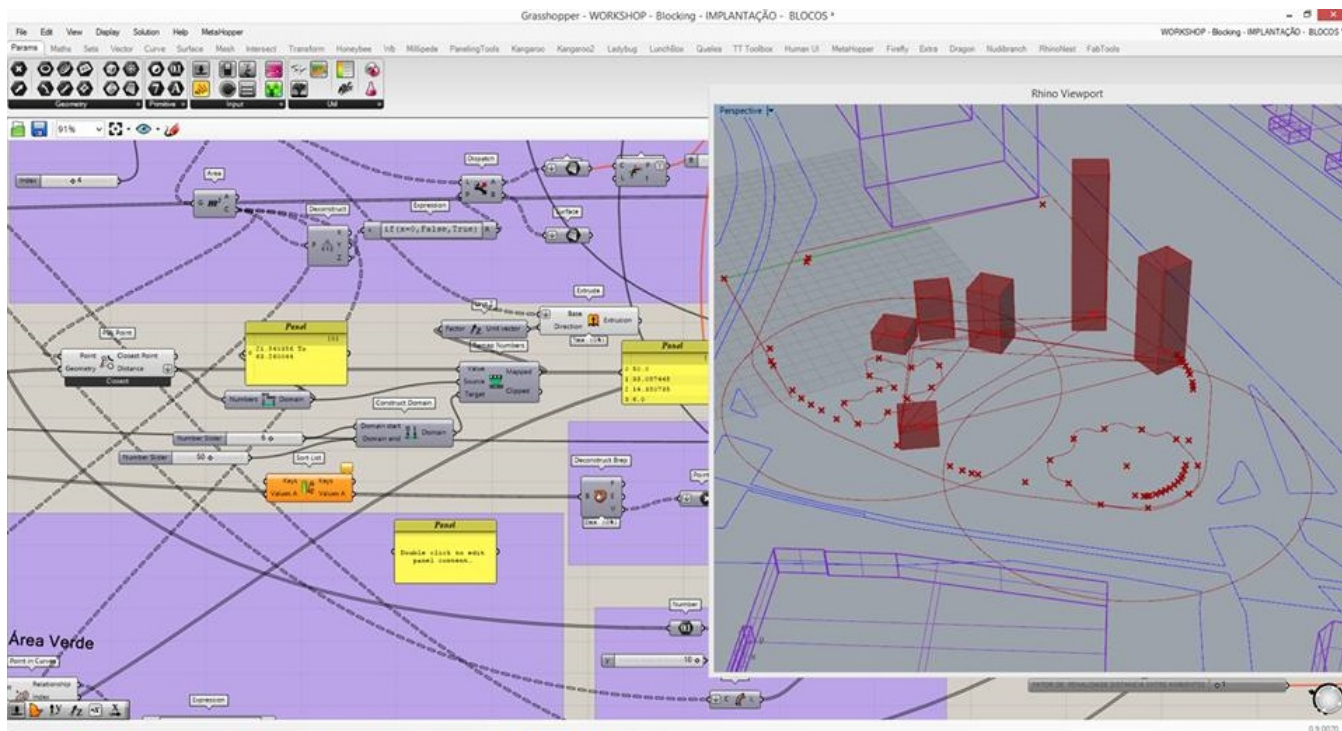
Grupo 3

Participantes: Dyego Digilandomenico e Rogério Lima.

O grupo 3 participou remotamente via *Google Hangout*. Este grupo buscou trabalhar com as temáticas das áreas verdes e do sombreamento. O grupo desenvolveu dois trechos de códigos que foram somados ao algoritmo da implantação tridimensional com blocos modulares.

A proposta do grupo agregou às áreas verdes penalidades de sobreposição e determinou que a solução analisaria as áreas de sobreposição de sombra entre os módulos da própria solução.

Figura 144 – Panorama geral do experimento do grupo 4 - Adriano Lima, Rodrigo Makert e Vinicius Gonçalves



Fonte: Do autor

7.4 Discussão e Questionário

Após o Workshop foi enviado aos alunos um questionário (que pode ser encontrado nos capítulos de apêndices deste trabalho), em que foram feitas perguntas referentes a sua experiência em usar algum sistema automatizado para a geração de layouts, quão relevante o método poderia ser no dia a dia de um escritório de arquitetura, o que gostaria de acrescentar ao sistema e quais foram as principais dificuldades durante o Workshop.

Foi sugerido que os participantes colaborassem com o questionário que era composto por perguntas de respostas afirmativas ou negativas, como também repostas livres discursivas. Apenas três participantes responderam ao questionário. Nenhum deles havia utilizado algum método de geração automatizada de *layouts*. Todos os participantes que responderam ao questionário acharam que o método pode colaborar suficientemente na tomada de decisões em algumas situações.

Quando foi pedido para que comparassem o método tradicional de projeto com o método automatizado, apontando os benefícios e malefícios de cada método, as respostas foram unânimes quanto ao potencial de se explorar a potencialidade de objetivos multicritério de uma implantação, mas que se os parâmetros de otimização são definidos de uma maneira errônea o método pode gerar soluções que não são boas do ponto de vista

arquitetônico.

De acordo com as respostas do formulário, mesmo com o número baixo de participantes que responderam ao questionário, devido ao caráter livre e discursivo de algumas perguntas e dos resultados obtidos no workshop, foi possível perceber que o método de *Space Planning* tem um grande potencial a ser explorado pelos arquitetos e com uma possível aceitação ao primeiro contato.

Uma das sugestões de melhoramento foi inserir um código para que o método pudesse trabalhar dentro de um *grid* ortogonal e outra sugestão foi a de trabalhar partes do código através de *clusters*.

7.5 Considerações sobre o Workshop

A proposta deste trabalho foi validar de maneira qualitativa o método através de uma experiência prática, sendo que foge ao escopo deste trabalho de mestrado uma análise quantitativa/estatística de quanto o método automatizado poderia otimizar as decisões dos arquitetos através de uma comparação de dados.

Parte VII

Considerações Finais

8 Considerações Finais

O potencial de cálculo dos computadores continua dobrando a cada dois anos. De acordo com Kaku (2012), a previsão que a lei de Moore só irá começar a desacelerar a partir de 2020, e no decorrer dos anos os transistores ficarão tão pequenos que atingirão o limite físico de um átomo (KAKU, 2012). Soluções para estes problemas já estão sendo exploradas através de computadores quânticos.

Os métodos do projeto computacional devem buscar andar em paralelo com essas novas tecnologias, como também explorar esse avanço computacional contínuo, pois vivemos em um momento histórico em que todas as áreas do conhecimento e ofícios estão sendo informatizados e, com isto, estão impulsionando exponencialmente avanços em distintas áreas do conhecimento, como genética, robótica, nanotecnologia, inteligência artificial, entre outras.

O projeto computacional teve muitos avanços nos últimos anos devido às novas ferramentas de interface gráfica e ao aumento da capacidade de processamento dos computadores. A combinação destes dois fatores está popularizando algoritmos que anteriormente estavam restritos a um pequeno círculo de estudiosos pioneiros em utilizarem os métodos computacionais mais recentes para explorar possibilidades no campo da arquitetura e construção.

Space Planning em Arquitetura e Urbanismo é uma destas metodologias com um grande potencial a ser explorado por essas novas ferramentas que foram introduzidas no mercado como *Grasshoper*, *Dynamo*, *Sverchok* e *Generative Components*.

Esta pesquisa buscou fazer uma revisão da metodologia de *Space Planning*, através de uma abordagem específica dos algoritmos evolutivos, fazendo um levantamento do material publicado até hoje, servindo de suporte a próximas pesquisas que pretendam abordar o tema, e implementando um método específico dentro dessa abordagem (o **agrupamento**).

Foram desenvolvidos uma série de experimentos que posteriormente foram aplicados em um exercício prático através da realização de um workshop.

Os experimentos serviram para ter um entendimento do assunto e também poderão servir de passo inicial para futuras pesquisas que estudarão o tema, pois todos os algoritmos dos experimentos serão disponibilizados juntamente com o volume desta dissertação.

Fica evidente, pós workshop realizado, a potencialidade do método em auxiliar arquitetos em suas decisões do dia-a-dia. Possíveis aplicações futuras com a metodologia

do *Space Planning* sendo aplicada através de uma interface gráfica, poderá colaborar significativamente nos distintos métodos que a compõem, corroborando com a **hipótese** formulada no início desta dissertação.

Como desdobramentos futuros de pesquisa, o desenvolvimento de um *plug-in* comercial poderia viabilizar ainda mais a popularização do método entre os arquitetos. Para isto, fica como possibilidade futura uma pesquisa que explore outros métodos de *space planning*, desenvolvendo um *plug-in* ou aplicação que possa atender a distintas demandas através de distintos métodos de geração automatizada de *layouts*.

Como últimas considerações, pode-se afirmar que o método de geração automatizada de *layouts* em Arquitetura e Urbanismo, tem um grande potencial a ser explorado por futuras pesquisas ou como aplicação comercial. Esta pesquisa buscou fazer um panorama geral da técnica e explorou através de alguns experimentos a viabilidade do método, aplicando-os em uma situação prática através de um workshop. Espera-se que esta pesquisa venha a contribuir com futuras pesquisas que busquem utilizar a metodologia de geração automatizada de *layouts*.

Referências

- BÄCK, T.; SCHWEFEL, H.-P. Evolutionary computation: An overview. In: IEEE. *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. [S.l.], 1996. p. 20–29. Citado na página 66.
- BANERJEE, A.; QUIROZ, J. C.; LOUIS, S. J. A model of creative design using collaborative interactive genetic algorithms. In: *Design Computing and Cognition'08*. [S.l.]: Springer, 2008. p. 397–416. Citado 3 vezes nas páginas 110, 111 e 112.
- BANZHAF, W.; NORDIN, P.; KELLER, R. E.; FRANCONI, F. D. *Genetic programming: an introduction*. [S.l.]: Morgan Kaufmann San Francisco, 1998. v. 1. Citado 4 vezes nas páginas 54, 59, 69 e 70.
- BENTLEY, P. *Evolutionary design by Computers*. [S.l.]: Morgan Kaufmann, 1999. Citado 16 vezes nas páginas 31, 55, 57, 58, 59, 60, 61, 66, 67, 68, 69, 70, 72, 73, 74 e 81.
- COATES, P.; MAKRIS, D. Genetic programming and spatial morphogenesis. 1999. Citado 4 vezes nas páginas 78, 79, 80 e 81.
- DAMSKI, J. C.; GERO, J. S. An evolutionary approach to generating constraint-based space layout topologies. In: *CAAD futures 1997*. [S.l.]: Springer, 1997. p. 855–864. Citado 2 vezes nas páginas 120 e 121.
- DOULGERAKIS, A. *Genetic programming+ unfolding embryology in automated layout planning*. Tese (Doutorado) — University College London, 2007. Citado 3 vezes nas páginas 92, 93 e 127.
- DUTTA, K.; SARTHAK, S. Architectural space planning using evolutionary computing approaches: a review. *Artificial Intelligence Review*, Springer, v. 36, n. 4, p. 311–321, 2011. Citado na página 84.
- EASTMAN, C. M. Cognitive processes and ill-defined problems: A case study from design. In: *Proceedings of the International Joint Conference on Artificial Intelligence: IJCAI*. [S.l.: s.n.], 1969. v. 69, p. 669–690. Citado na página 38.
- EASTMAN, C. M. *Heuristic algorithms for automated space planning*. [S.l.]: Institute of Physical Planning, Carnegie-Mellon University, 1971. Citado na página 25.
- EASTMAN, C. M. Automated space planning. *Artificial intelligence*, Elsevier, v. 4, n. 1, p. 41–64, 1973. Citado 2 vezes nas páginas 25 e 36.
- ELEZKURTAJ, T.; FRANCK, G. Geometry and topology: a user-interface to artificial evolution in architectural design. In: *18th eCAADe conference proceedings*. [S.l.: s.n.], 2000. p. 309–12. Citado 2 vezes nas páginas 108 e 109.
- FEOFILOFF, P.; KOHAYAKAWA, Y.; WAKABAYASHI, Y. Uma introdução sucinta à teoria dos grafos. *Disponível em <http://www.ime.usp.br/~pf/teoriadosgrafos>*, 2011. Citado na página 44.

- FERNANDO, R. A. Space planning and preliminary design using artificial life. 2014. Citado 2 vezes nas páginas 112 e 113.
- FLACK, R. W.; ROSS, B. J. *Evolution of architectural floor plans*. [S.l.]: Springer, 2011. Citado 2 vezes nas páginas 93 e 127.
- FOGEL, D. B. Foundations of evolutionary computation. In: *Proceedings of the SPIE, Volume 6228*. [S.l.: s.n.], 2006. p. 622801. Citado 3 vezes nas páginas 53, 55 e 66.
- FOGEL, L. J. Autonomous automata. *Industrial Research*, v. 4, n. 2, p. 14–19, 1962. Citado na página 67.
- FRAZER, J. An evolutionary architecture. 1975. Citado na página 71.
- GABRIEL, P. H. R.; DELBEM, A. C. B. *Fundamentos de algoritmos evolutivos*. [S.l.]: ICMC-USP, 2008. Citado 7 vezes nas páginas 53, 54, 64, 65, 66, 67 e 69.
- GARCES-PEREZ, J.; SCHOENEFELD, D. A.; WAINWRIGHT, R. L. Solving facility layout problems using genetic programming. In: MIT PRESS. *Proceedings of the 1st annual conference on genetic programming*. [S.l.], 1996. p. 182–190. Citado 2 vezes nas páginas 91 e 92.
- GARZA, A. G. D. S.; MAHER, M. L. Evolving design layout cases to satisfy feng shui constraints. In: *Proceedings of the fourth conference on computer aided architectural design research in Asia. Shanghai, China*. [S.l.: s.n.], 1999. p. 115–24. Citado 2 vezes nas páginas 103 e 104.
- GERO, J. S.; KAZAKOV, V. A. Evolving design genes in space layout planning problems. *Artificial Intelligence in Engineering*, Elsevier, v. 12, n. 3, p. 163–176, 1997. Citado 3 vezes nas páginas 114, 115 e 125.
- GERO, J. S.; KAZAKOV, V. A. Learning and re-using information in space layout planning problems using genetic engineering. *Artificial Intelligence in Engineering*, Elsevier, v. 11, n. 3, p. 329–334, 1997. Citado 3 vezes nas páginas 83, 114 e 115.
- GIL, A. C. Como elaborar projetos de pesquisa. *São Paulo*, v. 5, p. 61, 2002. Citado na página 31.
- GOLDBERG, D. E.; HOLLAND, J. H. Genetic algorithms and machine learning. *Machine learning*, Springer, v. 3, n. 2, p. 95–99, 1988. Citado na página 64.
- GRASON, J. An approach to computerized space planning using graph theory. In: ACM. *Proceedings of the 8th Design Automation Workshop*. [S.l.], 1971. p. 170–178. Citado 3 vezes nas páginas 25, 44 e 48.
- GUILFORD, J. P. The nature of human intelligence. McGraw-Hill, 1967. Citado na página 28.
- HARMANANI, H. M.; ZOUEIN, P. P.; HAJAR, A. M. A parallel genetic algorithm for the geometrically constrained site layout problem with unequal-size facilities. *International Journal of Computational Intelligence and Applications*, World Scientific, v. 4, n. 04, p. 375–400, 2004. Citado 2 vezes nas páginas 109 e 110.

- HERAGU, S. S. *Facilities design*. [S.l.]: CRC Press, 2008. Citado 5 vezes nas páginas 39, 40, 41, 49 e 50.
- HIREGOUDAR, C. *Facility Planning and Layout Design*. [S.l.]: Technical Publications, 2007. Citado 2 vezes nas páginas 42 e 43.
- HOMAYOUNI, H. *A genetic algorithm approach to space layout planning optimization*. Tese (Doutorado) — University of Washington, 2007. Citado 6 vezes nas páginas 25, 84, 115, 116, 117 e 127.
- JAGIELSKI, R.; GERO, J. S. A genetic programming approach to the space layout planning problem. In: *CAAD futures 1997*. [S.l.]: Springer, 1997. p. 875–884. Citado 4 vezes nas páginas 83, 99, 100 e 101.
- JO, J. H.; GERO, J. S. Representation and use of design knowledge in evolutionary design. *CAAD Futures '95. Singapore*, 1995. Citado 7 vezes nas páginas 94, 95, 96, 97, 98, 99 e 114.
- JO, J. H.; GERO, J. S. Space layout planning using an evolutionary approach. *Artificial Intelligence in Engineering*, Elsevier, v. 12, n. 3, p. 149–162, 1996. Citado na página 83.
- JONG, K. A. D. *Evolutionary computation: a unified approach*. [S.l.]: MIT press, 2006. Citado 3 vezes nas páginas 53, 54 e 66.
- JONG, K. D.; FOGEL, D. B.; SCHWEFEL, H.-P. A2. 3 a history of evolutionary computation. 1997. Citado na página 59.
- KADO, K. *An investigation of genetic algorithms for facility layout problems*. Tese (Doutorado) — University of Edinburgh, 1995. Citado na página 91.
- KADO, K. *An investigation of genetic algorithms for facility layout problems*. Tese (Doutorado) — University of Edinburgh, 1996. Citado na página 50.
- KAKU, M. *Physics of the future: How science will shape human destiny and our daily lives by the year 2100*. [S.l.]: Anchor, 2012. Citado na página 173.
- KNECHT, K.; KOENIG, R. Generating floor plan layouts with kd trees and evolutionary algorithms. In: *Generative Art Conf*. [S.l.: s.n.], 2010. p. 238–253. Citado 3 vezes nas páginas 121, 122 e 127.
- KOCHHAR, J. S.; FOSTER, B. T.; HERAGU, S. S. Hope: a genetic algorithm for the unequal area facility layout problem. *Computers & Operations Research*, Elsevier, v. 25, n. 7, p. 583–594, 1998. Citado 2 vezes nas páginas 102 e 127.
- KOOPMANS, T. C.; BECKMANN, M. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, JSTOR, p. 53–76, 1957. Citado na página 39.
- KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. [S.l.]: MIT press, 1992. v. 1. Citado 2 vezes nas páginas 69 e 90.
- KUSIAK, A.; HERAGU, S. S. The facility layout problem. *European Journal of operational research*, Elsevier, v. 29, n. 3, p. 229–251, 1987. Citado na página 49.

- LI, H.; LOVE, P. E. Site-level facilities layout using genetic algorithms. *Journal of Computing in Civil Engineering*, American Society of Civil Engineers, v. 12, n. 4, p. 227–231, 1998. Citado na página 102.
- LIGGETT, R. S. Automated facilities layout: past, present and future. *Automation in construction*, Elsevier, v. 9, n. 2, p. 197–215, 2000. Citado 8 vezes nas páginas 36, 37, 42, 49, 50, 84, 94 e 109.
- LIGGETT, R. S.; MITCHELL, W. J. Optimal space planning in practice. *Computer-Aided Design*, Elsevier, v. 13, n. 5, p. 277–288, 1981. Citado 5 vezes nas páginas 25, 97, 99, 101 e 114.
- MARTINO, J. *Algoritmos evolutivos como método para desenvolvimento de projetos de arquitetura*. Tese (Doutorado) — Universidade Estadual de Campinas, 2015. Citado 10 vezes nas páginas 31, 56, 60, 71, 73, 74, 75, 76, 77 e 128.
- MAYR, E. *O que é a evolução*. [S.l.]: Rocco, 2009. Citado na página 56.
- MICHALEWICZ, Z. *Genetic algorithms+ data structures= evolution programs*. [S.l.]: Springer Science & Business Media, 2013. Citado 2 vezes nas páginas 58 e 59.
- MILLER, W. R. Computer-aided space planning. In: ACM. *Proceedings of the 7th Design Automation Workshop*. [S.l.], 1970. p. 28–34. Citado 4 vezes nas páginas 25, 44, 45 e 46.
- MITCHELL, M. *An introduction to genetic algorithms*. [S.l.]: MIT press, 1998. Citado 3 vezes nas páginas 62, 63 e 64.
- MITCHELL, W. J. The theoretical foundation of computer-aided architectural design. *Environment and Planning B*, v. 2, n. 2, p. 127–150, 1975. Citado 3 vezes nas páginas 25, 37 e 41.
- MITCHELL, W. J. *A lógica da arquitetura: projeto, computação e cognição*. [S.l.]: Unicamp, 1990. Citado 2 vezes nas páginas 38 e 39.
- NEGROPONTE, N. *The architecture machine: towards a more human environment*. [S.l.]: MIT press Cambridge, MA, 1970. Citado na página 71.
- NOURIAN, P.; REZVANI, S.; SARIYILDIZ, S. Designing with space syntax. 2013. Citado na página 149.
- OTTEN, R. H. Automatic floorplan design. In: IEEE PRESS. *Proceedings of the 19th Design Automation Conference*. [S.l.], 1982. p. 261–267. Citado na página 88.
- OXMAN, R. Theory and design in the first digital age. *Design studies*, Elsevier, v. 27, n. 3, p. 229–265, 2006. Citado na página 26.
- POBLETE, C. I. de la B. Algoritmos evolutivos como modelo propositivo de diseño. *De Arquitectura*, n. 15, p. Pág-45, 2007. Citado 6 vezes nas páginas 61, 62, 63, 64, 65 e 66.
- RECHENBERG, I. Cybernetic solution path of an experimental problem. Ministry of Aviation, Royal Aircraft Establishment, 1965. Citado na página 66.
- REITMAN, W. R. Cognition and thought: an information processing approach. Wiley, 1965. Citado na página 38.

- ROCHA, A. J. M. *Architecture theory, 1960-1980: emergence of a computational perspective*. Tese (Doutorado) — Massachusetts Institute of Technology, 2004. Citado 2 vezes nas páginas 25 e 26.
- RODRIGUES, E.; GASPAR, A. R.; GOMES, Á. An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, part 1: Methodology. *Computer-Aided Design*, Elsevier, v. 45, n. 5, p. 887–897, 2013. Citado 4 vezes nas páginas 104, 105, 106 e 125.
- RODRIGUES, E.; GASPAR, A. R.; GOMES, Á. An evolutionary strategy enhanced with a local search technique for the space allocation problem in architecture, part 2: Validation and performance tests. *Computer-Aided Design*, Elsevier, v. 45, n. 5, p. 898–910, 2013. Citado 3 vezes nas páginas 104, 105 e 125.
- ROSENMAN, M.; GERO, J. Evolving designs by generating useful complex gene structures. *Evolutionary Design by Computers*, San Francisco, CA: Morgan Kaufmann, p. 345–364, 1999. Citado 2 vezes nas páginas 107 e 108.
- ROTH, J.; HASHIMSHONY, R. Algorithms in graph theory and their use for solving problems in architectural design. *computer-aided design*, Elsevier, v. 20, n. 7, p. 373–381, 1988. Citado na página 149.
- RUSSELL, S.; NORVIG, P.; INTELLIGENCE, A. A modern approach. *Artificial Intelligence*. Prentice-Hall, Egnlewood Cliffs, Citeseer, v. 25, 1995. Citado na página 25.
- SCHWEFEL, H.-P. Kybernetische evolution als strategie der experimentellen forschung in der strömungstechnik. *Master's thesis, Technical University of Berlin*, 1965. Citado na página 66.
- SHIFFMAN, D.; FRY, S.; MARSH, Z. *The nature of code*. [S.l.]: D. Shiffman, 2012. Citado na página 54.
- SIMON, H. A. The functional equivalence of problem solving skills. *Cognitive Psychology*, Elsevier, v. 7, n. 2, p. 268–288, 1975. Citado na página 38.
- SINGH, S. P.; SHARMA, R. R. A review of different approaches to the facility layout problems. *The International Journal of Advanced Manufacturing Technology*, Springer, v. 30, n. 5-6, p. 425–433, 2006. Citado na página 36.
- STEADMAN, P. *Architectural morphology: An introduction to the geometry of building plans*. [S.l.]: Taylor & Francis, 1983. Citado na página 149.
- TAM, K. Y. Genetic algorithms, function optimization, and facility layout design. *European Journal of Operational Research*, Elsevier, v. 63, n. 2, p. 322–346, 1992. Citado 4 vezes nas páginas 83, 88, 89 e 90.
- TAVAKKOLI-MOGHADDAIN, R.; SHAYAN, E. Facilities layout design by genetic algorithms. *Computers & industrial engineering*, Elsevier, v. 35, n. 3, p. 527–530, 1998. Citado na página 102.
- TERZIDIS, K. *Algorithmic architecture*. [S.l.]: Routledge, 2006. Citado na página 26.

THAKUR, M. K.; KUMARI, M.; DAS, M. Architectural layout planning using genetic algorithms. In: IEEE. *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*. [S.l.], 2010. v. 4, p. 5–11. Citado na página 119.

UHR, L. M. *Pattern recognition, learning, and thought: computer-programmed models of higher mental processes*. [S.l.]: Prentice-Hall Englewood Cliffs, NJ, 1973. Citado na página 38.

WONG, S. S.; CHAN, K. C. Evoarch: An evolutionary algorithm for architectural layout design. *Computer-Aided Design*, Elsevier, v. 41, n. 9, p. 649–667, 2009. Citado 2 vezes nas páginas 118 e 119.

ZUBEN, F. J. V. Computação evolutiva: uma abordagem pragmática. *Anais da I Jornada de Estudos em Computação de Piracicaba e Região (1a JECOMP)*, v. 1, p. 25–45, 2000. Citado na página 58.

Apêndices

APÊNDICE A – Questionário

3 respostas

[Visualizar todas as respostas](#)[Publicar análise](#)

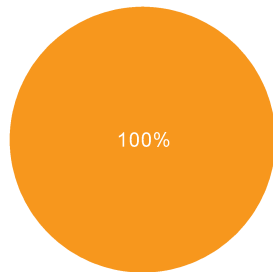
Resumo

Você já havia utilizado algum sistema de leiaute automatizado? Qual ?

Não

Não havia utilizado nenhum sistema de LAYOUT automatizado.

Você acha que o método pode colaborar (quanto) na tomada de decisões no dia-a-dia de um escritório de arquitetura ?



Nem um pouco 0 0%

Pouco 0 0%

Suficientemente, em algumas situações 3 100%

Bastante, quase em todas as situações 0 0%

Outros 0 0%

Comparando o método tradicional de se elaborar um plano de massas em arquitetura com o método automatizado, de acordo com o seu entendimento , cite quais são os benefícios e malefícios de se usar o método automatizado.

permite explorar soluções muticriteriais, onde não é criada uma forma, mas sim relações que geram a forma. Porem, se não utilizado do modo correto, pode gerar soluções que não são boas do ponto de vista arquitetônico.

Acredito que contribui por um estudo mais rápido e preciso, com diferentes variáveis e parâmetros.

Procura de soluções automática de acordo com critérios, previsão e estimativa de resultados, melhoria de processo de projeto e simulação

Liste se tem algo você mudaria ou acrescentaria ao método de geração automatizada

possibilidade de encaixar ou não as formas em grid ortogonal

Por ser um sistema aberto, acredito que acrescentaria conforme as necessidades do projeto.

Sugiro apresentar no início do workshop a definição sintética do algoritmo, tentando clarificar os fatores que estão envolvidos e como. Penso que é de equacionar apresentar o exercício simplificado, utilizando por exemplo cluster para inserir algumas das operações. A explicação completa dos componentes foi útil, mas como é muita informação pode dispersar os participantes. Se for possível ter um IP dedicado ou um programa mais fiável de transmissão eletrônica ou tele-transferência pode ajudar.

Descreva quais foram as suas principais dificuldades no decorrer do workshop

Não houveram na verdade dificuldades no workshop. Foi necessário entender o código e como usar alguns plug-ins, mas depois de aprendido, não ocorreram problemas. O fator mais complicado foi dosar corretamente as penalidades no fim do processo, pois quando se tem muitos objetivos com o mesmo peso, a solução parece não atender de modo correto nenhum deles.

o tempo poderia ser maior, mas foi possível compreender a ideia que foi abordada.

ver resposta inicial de síntese

Liste se tem algo que você mudaria ou acrescentaria à interface do algoritmo usado no workshop

Botão de "avanzar" e "voltar" ao invés do "ok" e que cada janela feche automaticamente ao clicar no botão avançar. Outras questões como definir usos e cores foram desenvolvidos no workshop

deixaria um pouco mais sugestivo aonde poderiam ser adicionado as novas variáveis.

ver resposta inicial de síntese

Liste se tem algo que você mudaria ou acrescentaria ao código do algoritmo usado no workshop

apesar de demandar maior processamento, faria a contagem de pontos de intersecção de volumes levando em conta o volume todo, não só as faces.

adicionaria um script com a projeção dos ambientes e um código que permitisse algum alinhamento entre eles

ver resposta inicial de síntese

Alguma sugestão ou opinião geral final ?

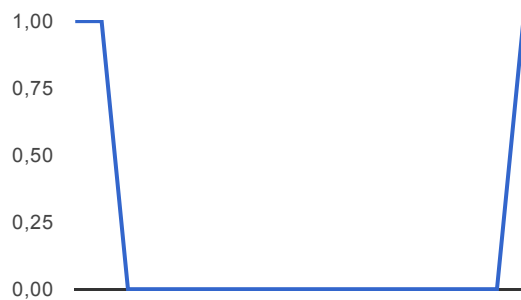
o componente final poderia ser dado numa disciplina de projeto dos primeiros anos, para que eles vejam esta possibilidade de geração automática.

Mesmo rápido, o workshop foi muito proveitoso e parabéns pelo trabalho. Sou de Campo Grande/MS e estudo arquitetura e urbanismo na UFMS, faço parte do grupo Algo+Ritmo que estuda diversos processos digitais. Seria uma honra se um dia aparecesse por aqui para uma conversa. Obrigado.

Gostaria de dar os parabéns pela iniciativa. É uma proposta inovadora pelo tema e pela metodologia de workshop presencial/on-line, o que certamente exigiu um trabalho de

preparação substancial .Penso que é de destacar a partilha do código e discussão do algoritmo, o que pode permitir uma evolução mais sustentada e rápida.

Número de respostas diárias



APÊNDICE B – Cronograma do Workshop

Programação

Workshop Space Planning Architecture

**Geração automatizada de leiautes
através de algoritmos evolutivos**

DIA – 20

9:00 – 10:00 : Recepção e instalação dos plug-ins
10:00 – 10:30 : Apresentação do workshop e dos conceitos básicos sobre SP
10:30 – 12:00 : Explicação do funcionamento dos algoritmos de SP
12:00 – 13:00 : Apresentação do exercício do workshop e formação dos grupos

Almoço

14:00 – 17:00 : Desenvolvimento do exercício
17:00 – 17:40 : Apresentação dos resultados
17:40 – 18:00 : Discussão e encerramento do Workshop

APÊNDICE C – Proposta do exercício do Workshop

Space Planning Architecture – Exercício

A proposta para o exercício de SP será a criação de um estudo de plano de massas para um complexo multifuncional localizado no bairro da Barra Funda em São Paulo. O estudo deve considerar a legislação urbana explícita no algoritmo de apoio, como também o programa de necessidades. O partido adotado na intervenção deve utilizar o algoritmo base de *space planning*, bem como incrementar outras estratégias de projeto à somatória da função *fitness* do Galápagos, de acordo com as condicionantes locais ou temáticas de projeto.

Como ponto de partida são dadas algumas condicionantes evidentes do local :

- A linha do Trem
- O viaduto Antártica
- As árvores do terreno
 - Os acessos
- Os edifícios do entorno

Os grupos devem escolher ao menos uma das condicionantes evidentes e trabalhar para otimizar um aspecto do edifício (vistas, sombras, insolação ,etc) que ficará a critério do grupo, de acordo com a estratégia de projeto adotada.



O local

$$\text{Área do Terreno} = 14.647 \text{ m}^2$$

$$\text{Taxa de Ocupação} = 0.5$$

$$\text{Coeficiente de Aproveitamento} = 2$$

Recuos = 5 metros

Programa de Necessidades - Complexo Multifuncional

Comércio: no máximo **30% da área total computável**

Módulos comerciais

Escritórios : máximo **30 % da área total computável.**

Recepção e controle de acesso no térreo

Habitação: máximo **40 % da área total computável.**

Recepção e controle de acesso no térreo
Áreas de lazer e convívio com dimensionamento proporcional
ao número de unidades.

Tempo de desenvolvimento da proposta **3 horas**

Serão **40 minutos** de apresentação das propostas divididos entre os grupos