UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Geração de Conteúdo Procedural por Busca Inovativa em Nichos

Alexandre Santos Melotti

Itajubá, outubro de 2016

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Alexandre Santos Melotti

Geração de Conteúdo Procedural por Busca Inovativa em Nichos

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação

Área de Concentração: Matemática da Computação

Orientador: DSc. Carlos Henrique Valério de Moraes

Outubro de 2016
Itajubá - MG

Agradecimentos

Ao meu professor e orientador Prof. Carlos Henrique Valério de Moraes, pelo apoio, sugestões, críticas construtivas e excelente exemplo profissional.

Em especial aos meus queridos pais e minha irmã, os quais foram responsáveis pela minha formação e educação, sempre presentes na minha vida.

À minha tia e professora de inglês Joana, cujos ensinamentos sempre me auxiliaram e me permitiram abrir diversas portas.

Aos meus excelentes amigos, com os quais sempre pude contar.

À CAPES pelo financiamento através da bolsa de estudos.

You can discover more about a person in an hour of play than in a year of conversation.

Plato

Resumo

Com a crescente demanda de ambientes virtuais diferenciados surge a necessidade da geração dinâmica de conteúdo, agilizando o desenvolvimento, reduzindo custos de produção e ampliando a vida útil do produto. Em termos gerais, mundos procedurais geram uma gama maior de detalhamento no ambiente e personalizam a experiência, sem prejudicar a performance do sistema, mas isso só é garantido quando a qualidade do conteúdo obtido satisfaz o usuário em qualidade e diversidade. Desta forma, é necessária a utilização de algoritmos que sejam capazes de garantir a qualidade da experiência gerada sem faltar com a diversidade. Este trabalho apresenta uma técnica que pode ser utilizada no desenvolvimento de ambientes virtuais tanto em sua produção, oferecendo ao desenvolvedor possibilidades distintas de design para serem elaboradas e refinadas, como também durante a execução, construindo simulações únicas enquanto sendo utilizada pelo usuário. Assim, a técnica desenvolvida utiliza nichos morfológicos para garantir diversidade de soluções, produzindo resultados que apresentam as melhores soluções dentro de cada grupo, superando em qualidade as técnicas existentes como o busca inovativa e competição local.

Abstract

With the increasing demand for differentiated virtual environments, the need for dynamic content generation arises, speeding up development, reducing productions costs and increase the product shelf life. In general, procedurally generated worlds, offer a greater range of details in the environment and experience customization without sacrificing performance, but this is only guaranteed when the quality of the content obtained satisfies the user in quality and diversity. Thus, the use of algorithms that are able to guarantee the quality of the generated experience without lacking diversity are required. This paper presents a technique that can be used to develop virtual environments both in its production, providing the developer with different design possibilities to be developed and refined, as well as execution, building unique simulations while interfacing with the user. Therefore, the technique developed makes use of morphological niches to ensure diversity of solutions, producing results that provide the best solutions within each group, surpassing in quality the existing techniques such as novelty search and local competition.

Lista de Figuras

1 Captura de tela do jogo *Rogue* (TOY et al., 1980). Um dos primeiros jogos a criar mapas de forma procedural, predecessor que deu nome ao gênero de jogos conhecidos como *roguelike*. . . . . p. 14

2 Captura de tela do jogo *Diablo* (BLIZZARD NORTH, 1996). Os jogos da franquia utilizam técnicas de PCG para criar itens e mapas, fazendo com que jogadores retornem ao jogo pela possibilidade de encontrar melhores itens para seus personagens. . . . . p. 15

3 Captura de tela do jogo *Minecraft* (MOJANG, 2011). Cidade criada por jogadores em um mundo gerado completamente de forma procedural. Esta combinação é uma excelente fómula para criar um senso de propriedade e autoria em seus jogadores. . . . . p. 16

4 Captura de tela do jogo *Oblivion* (BETHESTA GAME STUDIOS, 2006). Tecnicas de *offline* PCG foram utilizadas para geração da maioria de seus terrenos e florestas. . . . . p. 17

5 Captura de tela do jogo *Borderlands* (GEARBIX SOFTWARE, 2009). Tecnicas de *online* PCG foram utilizadas para geração de armas do jogo, produzindo mais de 3 milhões de combinações possíveis. . . . . p. 18

6 Captura de tela do jogo *Spelunky* (YU, 2012). . . . . p. 19

7 A tela inicial de quatro níveis distintos criados por um mesmo gerador procedural em (KERSSEMAKERS et al., 2012). . . . . p. 22

8 Exemplos de armas criadas para o jogo *Galactic Arms Race* com a técnica descrita em (HASTINGS; GUHA; STANLEY, 2009b). . . . . p. 23

9 Exemplos de flores criadas com fractais através do processo colaborativo de jogadores em (RISI et al., 2012). . . . . p. 24

10 Exemplos de mapas evoluídos com o uso de diferentes funções de avaliação em (LIAPIS; YANNAKAKIS; TOGELIUS, 2013b). . . . . p. 25

11 Tela de seleção da ferramenta implementada em (LIAPIS; YANNAKAKIS; TOGELIUS, 2013c) onde o ator escolhe um mapa de maior resolução para prosseguir. . . . . p. 28

12 Exemplo de criaturas virtuais com capacidade de locomoção razoável e diferentes morfologias encontradas em uma única execução do algoritmo apresentado em (LEHMAN; STANLEY, 2011b). . . . . p. 29

13 Da esquerda para a direita, exemplos de mapas criados pelas representações cromática, positiva e negativa em (ASHLOCK; LEE; MCGUINNESS, 2011). . . . . p. 31

14 Exemplos de mapas gerados através da técnica proposta em (VALTCHANOV; BROWN, 2012). . . . . p. 32

15 Exemplos de mapas criados pelas diferentes representações propostas em (CARDAMONE et al., 2011). . . . . p. 33

16 Exemplos de mapas gerados em (LANZI; LOIACONO; STUCCHI, 2014) que utilizam somente a representação negativa proposta em (CARDAMONE et al., 2011). . . . . p. 33

17 Diagrama de Atividades de um típico Algoritmo Genético. . . . . p. 36

18 Diagrama de Atividades de um Algoritmo Genético de Busca Inovativa. . p. 36

19 P1 está em uma região mais densa que P2. . . . . p. 37

20 Diagrama de Atividades do Algoritmo Genético de Busca Inovativa com Critério Mínimo. . . . . p. 39

21 Frentes não dominadas em um problema multiobjetivo. . . . . p. 42

22 Representação do cuboide utilizado para o cálculo de distância de aglomeração. . . . . p. 43

23 Diagrama de Atividades para o cálculo de Competição Local em um sistema de Busca Inovativa. . . . . p. 46

24 Indivíduos em uma mesma frente não dominada são selecionados por critério de Dispersão. . . . . p. 50

25 Diagrama de Atividades para o método proposto. . . . . p. 51

26 Representação de um fenótipo de mapa um 16 x 16. . . . . p. 53

27 Estrutura da representação genotípica de um mapa. . . . . p. 53

28 Representação das possíveis movimentações do jogador: em verde movimentações com peso normal, em amarelo movimentações com peso maior e em vermelho movimentações não permitidas. . . . . p. 55

29 Mapa obtido com Nota de Avaliação 157. . . . . p. 55

30 Diagrama de Classes do Pacote Computacional. . . . . p. 58

31 Diagrama de Classes do pacote MapEvolution, extendendo as classes abstratas do pacote GenericImplementation. . . . . p. 62

32 Gráfico dos valores obtidos para Nota Local Média e Máxima. . . . . p. 68

33 Gráfico da quantidade de indivíduos em C_f. . . . . p. 69

34 Gráfico das Notas de Avaliação Média obtidas por cada configuração em P_f e C_f. . . . . p. 70

35 Gráfico da Dispersão Média obtidas por cada configuração em P_f e C_f. . p. 72

36 Gráfico de Nota de Avaliação Média e Dispersão Média para os conjuntos C_f obtidos em cada uma das configurações baseadas na NS. . . . . p. 73

Lista de Tabelas

1 Parâmetros de execução utilizados por cada configuração. . . . . p. 65

2 Valores obtidos por cada configuração para Nota Local. . . . . p. 67

3 Valores obtidos por cada configuração para a Quantidade de indivíduos em C_f. . . . . p. 69

4 Valores obtidos por cada configuração para Nota de Avaliação Média em P_f e C_f. . . . . p. 70

5 Valores obtidos por cada configuração para Dispersão Média em P_f e C_f. p. 71

6 Tempos e consumo de memória em cada execução obtido pelas configurações. . . . . p. 74

7 Tempo médio de execução por solução obtida em cada configuração. . . . p. 74

Glossário

AG	*Algoritmo Genético*
EDPCG	*Experience-Driven Procedural Content Generation*
MCNS	*Minimal Criteria Novelty Search*
MOEA	*Multiobjective Evolutionary Algorithm*
NS	*Novelty Search*
NSGA-II	*Nondominated Sorting Genetic Algorithm II*
PCG	*Procedural Content Generation*
PMCNS	*Progressive Minimal Criteria Novelty Search*
SBPCG	*Search-Based Procedural Content Generation*

Sumário

1 Introdução p. 13

1.1 Contexto . . . . . p. 13

1.2 Objetivos . . . . . p. 20

1.3 Revisão Bibliográfica . . . . . p. 21

1.4 Estrutura da Dissertação . . . . . p. 34

2 Metodologia p. 35

2.1 Novelty Search . . . . . p. 35

2.1.1 Minimal Criteria Novelty Search . . . . . p. 38

2.1.2 Progressive Minimal Criteria Novelty Search . . . . . p. 39

2.2 NSGA-II . . . . . p. 41

2.3 Diversificação Morfológica . . . . . p. 44

2.3.1 Balanceando Performance e Inovação . . . . . p. 44

2.3.2 Competição Local . . . . . p. 45

3 Desenvolvimento p. 47

3.1 Estratégia de Competição Local . . . . . p. 47

3.2 Critério Mínimo Global . . . . . p. 48

3.3 Estratégia de Seleção . . . . . p. 49

3.4 Conjunto de Soluções . . . . . p. 50

3.5 Metodologia Proposta . . . . . p. 51

3.6 Evoluindo Mapas para um Jogo de Aventura . . . . . p. 51

3.6.1 Representação do Mapa . . . . . p. 52
3.6.2 Função de Avaliação . . . . . p. 54
3.6.3 Função Distância . . . . . p. 56
3.7 Diagrama de Classes . . . . . p. 57
3.7.1 Pacote AForge.Genetic . . . . . p. 59
3.7.2 Pacote MultiObjectiveGA . . . . . p. 59
3.7.3 Pacote NSGAII . . . . . p. 60
3.7.4 Pacote NoveltySearch . . . . . p. 60
3.7.5 Pacote GenericImplementation . . . . . p. 61
3.7.6 Pacote MapEvolution . . . . . p. 61

4 Experimentos e Resultados p. 63

4.1 Parâmetros de Execução . . . . . p. 64
4.2 Métricas . . . . . p. 66
4.3 Resultados Obtidos para o Jogo Procedural . . . . . p. 67
4.3.1 Nota Local . . . . . p. 67
4.3.2 Quantidade de indivíduos em C_f . . . . . p. 68
4.3.3 Nota de Avaliação . . . . . p. 70
4.3.4 Dispersão . . . . . p. 71
4.3.5 Resultados Obtidos para C_f . . . . . p. 72
4.4 Tempo de Execução . . . . . p. 73
4.5 Discussão das Análises e Testes . . . . . p. 75

5 Conclusão p. 76

5.1 Trabalhos Futuros . . . . . p. 77

Referências p. 79

Apêndice A – Tabelas dos Valores Obtidos nos Testes p. 82

A.1 Algoritmo Genético . . . . . p. 83

A.2 Busca Inovativa . . . . . p. 85

A.3 Competição Local . . . . . p. 86

A.4 Proposta . . . . . p. 87

Apêndice B – Exemplos de Mapas Obtidos p. 88

B.1 Algoritmo Genético . . . . . p. 88

B.2 Busca Inovativa . . . . . p. 89

B.3 Competição Local . . . . . p. 90

B.4 Proposta . . . . . p. 91

1 Introdução

1.1 Contexto

Com o crescente mercado de jogos digitais, a demanda por conteúdos novos, cada vez mais sofisticados, diferenciados e muitas vezes personalizados, também aumenta. Infelizmente o custo e escalabilidade para criação manual de novos conteúdos, tão variados, é alta e não escalável. Com isso, muita atenção tem sido dada para técnicas de Geração Procedural de Conteúdo (*Procedural Content Generation* - PCG), tanto no meio comercial quanto acadêmico. PCG é o termo geral para o processo onde métodos algorítmicos são utilizados para geração de conteúdo, seja em jogos ou outros contextos (ANGELIDES; AGIUS, 2014). O conteúdo gerado pode incluir, mas não se limita a, terrenos, mapas, itens, recompensas, posicionamento de inimigos, distribuição de recursos, áudio, árvores de diálogo, objetivos de missões, padrões de comportamentos, e muitos outros. Estudos recentes, como (HENDRIKX et al., 2013), (TOGELIUS et al., 2011) e (YANNAKAKIS; TOGELIUS, 2011), avaliam e relacionam diversas metodologias e áreas onde a PCG pode ser aplicada em jogos.

De forma geral, sistemas PCG são desenvolvidos para produzir variação de conteúdo com base em permutações, e como na maioria das vezes se baseiam em aleatoriedade para gerar diversidade e escalabilidade, a não ser que sejam associados a uma semente aleatória específica, cada execução irá gerar um resultado diferente. Isso não só permite aos desenvolvedores criar uma grande variação de modelos e conteúdo, mas também possibilita que jogadores possam vivenciar diferentes desafios e eventos cada vez que jogam um mesmo jogo. De fato, técnicas de PCG são apropriadas para a solução de diversos problemas na área de desenvolvimento de jogos (ANGELIDES; AGIUS, 2014).

São elas:

- **Restrições de memória:** Embora tenha ganho muito mais atenção e visibilidade nos últimos tempos, várias técnicas de PCG vêm sendo utilizadas desde a década de 80 em jogos como *Rogue* (TOY et al., 1980) (Figura 1),

River Raid (ACTIVISION, 1982b), *Pitfall!* (ACTIVISION, 1982a) e *Elite* (BRABEN; BELL, 1984), por contribuir com soluções de problemas como restrições de memória e armazenamento (AYCOCK, 2016). Ainda que este problema possa ser encontrado em consoles portáteis e jogos de celular nos dias de hoje, a popularidade dos métodos de PCG atualmente se deve à outras adversidades, relacionadas à crescente complexidade em geração de novos conteúdos, que também podem ser resolvidas através de seu uso.

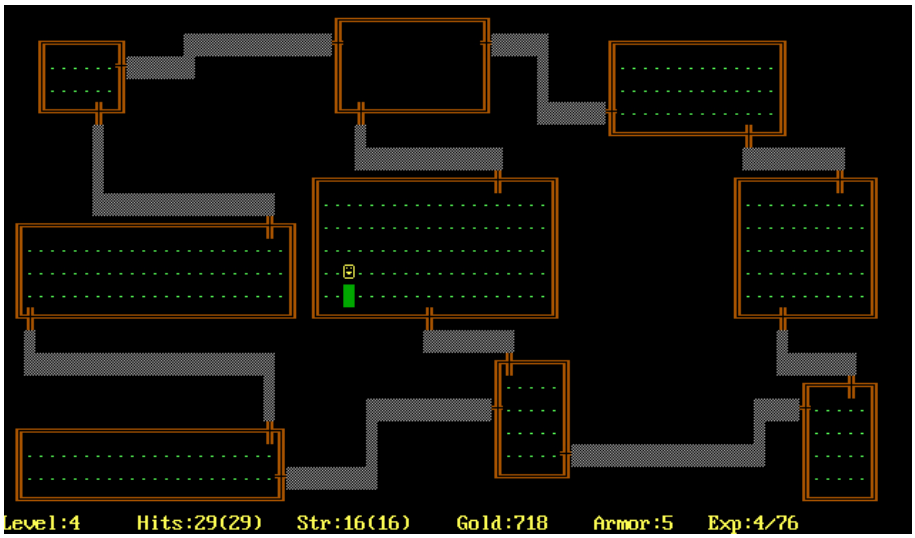


Figura 1: Captura de tela do jogo *Rogue* (TOY et al., 1980). Um dos primeiros jogos a criar mapas de forma procedural, predecessor que deu nome ao gênero de jogos conhecidos como *roguelike*.

- **Demanda por maior nível de detalhes:** Com jogos tendo mundos cada vez mais imersivos e complexos, a exigência por nível de detalhes e interação por parte dos jogadores aumenta de forma drástica. A criação manual de detalhes como folhas, rochas e outros artefatos em um mundo virtual complexo como os de jogos de hoje em dia é impraticável. O jogo *Left 4 Dead* (VALVE SOUTH, 2008), por exemplo, utiliza técnicas de PCG para geração de zumbis diferenciados utilizando permutações de partes de modelos como cabeças, tórax e braços de forma inteligente com base nos cenários.

- **Longevidade e valor de repetição (*replay value*):** Com a finalidade de aumentar a quantidade de conteúdo oferecida aos jogadores, e sem a necessidade de criar novos conteúdos manualmente. Alguns jogos como *Diablo* (BLIZZARD NORTH, 1996) (Figura 2), por exemplo, utilizam as características aleatórias da PCG para geração de itens e terrenos, fazendo com

que jogadores retornem a jogá-lo somente pela possibilidade de encontrar novos itens ou locais para suas aventuras. Em jogos como *Minecraft* (MOJANG, 2011) e *Terraria* (RE-LOGIC, 2011), por exemplo, a exploração de conteúdo gerado aleatoriamente, com base em elementos já conhecidos, é a principal mecânica do jogo.

Figura 2: Captura de tela do jogo *Diablo* (BLIZZARD NORTH, 1996). Os jogos da franquia utilizam técnicas de PCG para criar itens e mapas, fazendo com que jogadores retornem ao jogo pela possibilidade de encontrar melhores itens para seus personagens.

- **Novidade e inovação:** Também devido aos padrões de aleatoriedade oferecidos pela PCG, uma característica interessante emergente é a possibilidade de encontrar combinações inovadoras e inesperadas de conteúdo; estas que muitas vezes são capazes de surpreender até mesmo os próprios desenvolvedores. Enquanto em sistemas lineares a utilização de árvores de decisão pode aumentar a variabilidade das experiências dos jogadores, o custo para produção de diversas experiências também é crescente. A imprevisibilidade provida pela PCG oferece aos jogadores o senso de que cada mundo gerado é único e existe para um propósito próprio.

- **Propriedade e autoria:** Provavelmente uma consequência do item anterior, os jogadores sentem que, devido ao conteúdo único e diferenciado, a narrativa dos eventos ocorridos durante jogos com PCG se torna algo mais pessoal. Especialmente em jogos como *Minecraft* (MOJANG, 2011) (Figura

3), *Terraria* (RE-LOGIC, 2011) e *Spore* (MAXIS, 2008), a combinação de conteúdos únicos gerados pela PCG, em adição ao criado pelos próprios jogadores, oferece um forte senso de propriedade e autoria, engajando ainda mais o jogador ao mundo sendo desenvolvido.

Figura 3: Captura de tela do jogo *Minecraft* (MOJANG, 2011). Cidade criada por jogadores em um mundo gerado completamente de forma procedural. Esta combinação é uma excelente fómula para criar um senso de propriedade e autoria em seus jogadores.

É importante notar também, que técnicas de PCG podem ser executadas em modo *offline*, *online*, ou até mesmo como uma combinação de ambos. Conforme definido em (TOGELIUS et al., 2011), o termo *offline* PCG se refere a geração de conteúdo durante a fase de desenvolvimento da aplicação e, de forma geral, uma vez criado e publicado, o conteúdo nunca se altera; em contrapartida, *online* PCG descreve uma forma mais "ativa" de geração de conteúdo, sendo aplicada durante a execução do programa. A última, frequentemente envolve a geração de conteúdo apoiado por regras baseadas em estados atuais da aplicação ou usuário. Como forma de combinação entre as duas técnicas, podemos ter mapas imutáveis em um jogo que tenham sido criados de forma *offline*, e então usar técnicas *online* para população e variação de itens e inimigos no mapa. Outra forma possível para a combinação de ambos é em uma aplicação que ofereça conteúdo diversificado diariamente, com base em estatísticas e dados recolhidos do usuário nos últimos dias.

A técnica de *offline* PCG pode ser considerada como uma forma de auxílio à tomada de decisões, sendo utilizada para gerar esboços de conteúdos que serão refinados pelos desenvolvedores em etapas seguintes. Outra forma de utilização de *offline* PCG, é justamente para o auxílio a criação de detalhamento e diversidade durante o processo de

refinamento de conteúdo, gerando automaticamente vegetação, construções, personagens, entre outros, de forma abrangente e eficiente. Como exemplo de *offline* PCG, o jogo *Oblivion* (BETHESTA GAME STUDIOS, 2006) (Figura 4) utilizou de um sistema PCG assistido para a geração da maioria de seus terrenos e florestas.

Figura 4: Captura de tela do jogo *Oblivion* (BETHESTA GAME STUDIOS, 2006). Tecnicas de *offline* PCG foram utilizadas para geração da maioria de seus terrenos e florestas.

Online PCG em geral ocorre de forma mais reativa que sua contraparte, gerando uma relação interativa contínua com o atual estado do jogo. Esta modalidade de PCG é mais complexa em termos de definição e planejamento, uma vez que sua utilização permite que o mundo do jogo e suas mecânicas mantenham evoluções e mudanças constantes e de forma autônoma. Por este motivo, conteúdos gerados desta forma acabam influenciando as dinâmicas de jogo e muitas vezes o nível de dificuldade dos desafios enfrentados pelo jogador. O jogo *Borderlands* (GEARBIX SOFTWARE, 2009) (Figura 5) utiliza *online* PCG para combinar armas, tipos de munições, efeitos especiais, e modificações em atributos como alcance e outros, criando uma vasta combinação de armas possíveis; dependendo da arma adquirida pelo jogador, a dinâmica do jogo se altera completamente.

O controle sobre os efeitos do conteúdo gerado em PCG é tão complexo que vários estudos têm sido realizados na área. Uma nova ramificação de PCG, que se dedica a considerar as experiências de um ator humano para a geração de conteúdo, chamada Geração Procedural de Conteúdo Dirigida à Experiência (*Experience-Driven Procedural Content Generation* - EDPCG), é estudada e detalhada em (YANNAKAKIS; TOGELIUS,

Figura 5: Captura de tela do jogo *Borderlands* (GEARBIX SOFTWARE, 2009). Tecnicas de *online* PCG foram utilizadas para geração de armas do jogo, produzindo mais de 3 milhões de combinações possíveis.

2011). Na EDPCG, um ator humano (jogador) participa na seleção e evolução de novos conteúdos de forma direta ou indireta, desta forma o conteúdo gerado de forma online pode ser personalizado à cada indivíduo, suprindo suas preferências ou necessidades com base nos dados coletados sobre o mesmo. (HASTINGS; GUHA; STANLEY, 2009b) e (HASTINGS; GUHA; STANLEY, 2009a) utilizam um algoritmo evolutivo que se baseia nas preferências do jogador para evoluir diferentes tipos de armas em um jogo de ação e combate de aeronaves.

Como técnicas de *online* PCG são integradas e aplicadas durante a execução dos programas, alguns requisitos se fazem necessários na maioria dos casos: o algoritmo precisa ser executado de forma rápida, necessita ter um tempo de execução previsível e seus resultados devem ter qualidade esperada. Os dois primeiros requisitos, ambos relacionados ao tempo de execução do algoritmo, se devem ao fato de que o uso da *online* PCG muitas vezes ocorre durante a ação do jogo. Em jogos que exigem reflexos rápidos e precisos, gargalos de processamento são inaceitáveis, exigindo que tais técnicas sejam processadas de forma rápida e eficiente. Mesmo quando a PCG é utilizada em fases preparatórias, como em telas de carregamento entre níveis, embora seja aceitável por parte dos usuários aguardar algum tempo, uma espera muito grande se torna algo indesejável e incômodo, afetando inclusive a imersão do jogador no contexto do jogo.

Quanto a qualidade do conteúdo gerado, o problema surge quando alguns itens são criados de forma incompatível ou defeituosa. Enquanto alguns jogos como *Spelunky*

(YU, 2012) (Figura 6) se beneficiam pela geração de conteúdos incrivelmente difíceis, a alta dificuldade nestes casos é constante e faz parte de seu *design*; embora sejam extremamente complicados, todos os níveis possuem solução, nenhum conteúdo difícil é gerado por acaso, e a alta dificuldade por consequência do uso de *online* PCG é proposital. Em contraste, conteúdos mal formados podem influenciar a dinâmica do jogo de forma muito negativa. Se algum desafio gerado se torna impossível, ou há um grande declive entre os níveis de desafios gerados, o equilíbrio entre desafio e tédio é rompido e o jogo deixa de ser divertido.

Figura 6: Captura de tela do jogo *Spelunky* (YU, 2012).

Diversas formas de controle de qualidade de conteúdo gerado tem sido estudadas, (TOGELIUS et al., 2011) provê um estudo e taxonomia de pesquisas que utilizam algoritmos evolucionários e outras formas meta-heurísticas de algoritmos de busca para gerar conteúdos automaticamente em jogos, dando o nome a estes de Geração Procedural de Conteúdo Baseada em Busca (*Search-Based Procedural Content Generation* - SBPCG). A utilização de Algoritmos Genéticos para controle de qualidade de conteúdo é uma boa solução, mas pode apresentar alguns problemas em relação ao tempo de execução e à diversificação do conteúdo gerado. Dependendo da forma como o problema é modelado, e da natureza do espaço de busca, execuções podem levar um bom tempo para encontrar uma solução satisfatória. Além disso, normalmente os indivíduos obtidos na população final são muito semelhante, e não há garantia de que uma nova execução não irá convergir para o mesmo ponto. Isso pode resultar na falha em obtenção de conteúdo suficientemente diversificado, o que normalmente é uma das principais motivações para o uso de PCG nos jogos de hoje.

Analisando estes problemas, seria interessante um método que consiga gerar,

em uma só execução, conteúdo bom e diversificado, garantindo pelo menos um mínimo de qualidade funcional e diferenciação entre si. Tal método poderia ser utilizado tanto de forma *offline*, oferecendo ao desenvolvedor diversas possibilidades de *design* para serem trabalhadas em cima; como de forma *online* em fases preparatórias do jogo, como telas de carregamento, gerando de uma só vez vários itens para serem utilizados durante a execução. Todas estas qualidades em um só método poderiam não só garantir que o jogador não fique entediado e frustrado, como poupar tempo com execuções desnecessárias gerando conteúdos semelhantes e não funcionais. Fora do contexto de jogos, este método poderia ser de excelente aplicação em problemas de auxílio à tomada de decisão onde há a necessidade de sugestões diferenciadas.

1.2 Objetivos

O objetivo geral deste trabalho é a elaboração de uma solução evolutiva que seja capaz de obter um número razoável de indivíduos diferenciados, com qualidade funcional aceitável. Além disso é desejável que esta seja livre de interação humana e que possa ser utilizada tanto de forma *online* quanto *offline*. O contexto utilizado para desenvolvimento e testes será a geração procedural de fases para um jogo de aventura, onde é desejável encontrar mapas visualmente distintos e que possuam uma boa qualidade de navegação e exploração.

Sendo assim, este trabalho propõe o desenvolvimento de uma metodologia que evolua soluções distintas e de qualidade aceitável para sistemas com necessidade de resultados diferenciados.

Os objetivos específicos são:

- Elaborar a estrutura genética para a construção e evolução de mapas;
- Avaliar a qualidade funcional dos mapas através de uma função de avaliação;
- Classificar o nível de diferença entre mapas;
- Utilizar as qualidades funcionais obtida pelo Algoritmo Genético (AG) convencional;
- Utilizar as qualidades de diversidade introduzidas pela Busca Inovativa (*Novelty Search* - NS);

- Combinar de forma eficiente as qualidades oferecidas pelo AG convencional e pela NS;
- Obter uma quantidade satisfatória de soluções em uma única execução;
- Garantir que as soluções encontradas sejam abrangentes e aptas para a solução do problema.

1.3 Revisão Bibliográfica

Nesta seção estão apresentados trabalhos que tentam solucionar de diversas maneiras o problema identificando, avaliando seus pontos negativos e positivos. Serão apresentados ainda, trabalhos que possam servir como base ou inspiração para o desenvolvimento da solução ou contexto escolhido.

A identificação da semelhança de conteúdos gerados de forma automática, (KERSSEMAKERS et al., 2012) sugere a criação de um gerador procedural de geradores procedurais para fases de *Super Mario Bros.* (MIYAMOTO, 1985), na tentativa de automatizar o processo de variabilidade de conteúdo. Os geradores procedurais finais produzidos pelas técnicas utilizadas, são capazes de evoluir mapas com tema visual comum, sendo facilmente possível identificar as semelhanças entre mapas criados por um mesmo gerador procedural. A Figura 7 mostra a tela inicial de quatro níveis distintos criados por um mesmo gerador procedural. A proposta do autor é criar, de forma procedural, geradores procedurais com temas visuais diversos, sendo então capaz de usar diferentes geradores resultantes do processo evolutivo para gerar mapas distintos.

Com isso, é proposto a criação automática de geradores procedurais distintos com base em SBPCG e EDPCG. Neste processo, geradores procedurais são apresentados a um agente humano, que decide quais geradores serão combinados para formar os indivíduos da próxima geração. Com o resultado final, é possível se criar conjuntos de fases com temas distintos, podendo ser caracterizados como fases de "mundos" diferentes no universo de *Super Mario Bros.*.

Mesmo sendo capaz de gerar fases distintas, a solução se baseia no uso de um ator humano como principal fator de seleção. Por conta disso, este procedimento se torna inviável como solução para o problema proposto, pois é desejável uma solução que possa ser utilizada tanto de forma *offline* como *online*, sem que a imersão do jogador seja quebrada. Além disso, mesmo que o ator humano seja substituído por um agente

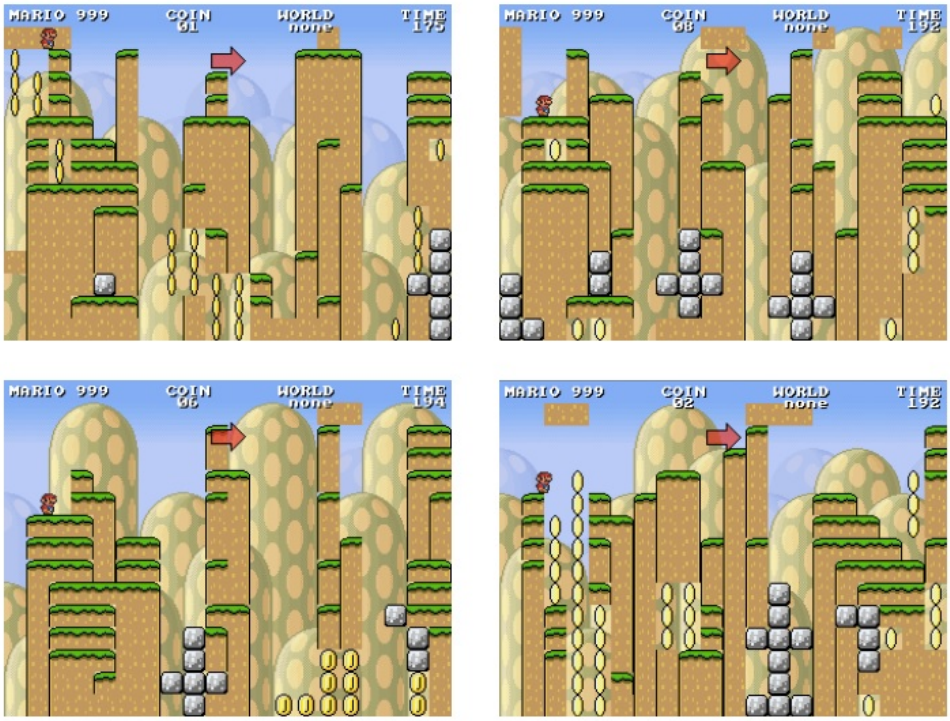


Figura 7: A tela inicial de quatro níveis distintos criados por um mesmo gerador procedural em (KERSSEMAKERS et al., 2012).

automatizado, o método não oferece nenhuma garantia de qualidade sobre o conteúdo gerado.

Em (HASTINGS; GUHA; STANLEY, 2009b) e (HASTINGS; GUHA; STANLEY, 2009a) são evoluídas diferentes armas com base nas preferências do jogador para um jogo de combate multi-jogador. A solução é capaz de gerar armas com efeitos visuais incrivelmente diferentes e foi utilizada comercialmente para o lançamento do jogo *Galactic Arms Race* (EVOLUTIONARY GAMES, 2012). A Figura 8 apresenta exemplos de armas geradas para o jogo *Galactic Arms Race* durante uma partida.

A solução proposta aqui também é uma mistura entre SBPCG e EDPCG, onde partículas visuais para os efeitos da armas são evoluídos com base em um algoritmo evolutivo e em escolhas e preferências dos jogadores. Durante a execução do jogo, cada item recebe uma nota de avaliação baseada na frequência em que aquele conteúdo é efetivamente utilizado. Novos itens são posicionados na fase para que os jogadores possam pegá-los, somente itens apanhados pelo jogador são elegíveis para reprodução. O algoritmo seleciona itens, dentre os já utilizados pelo jogador, para reprodução e geração de novos conteúdos. Os itens são selecionados de forma probabilística com base na nota de avaliação calculada. Para cada novo item adicionado à fase, há uma chance de que um item pré-

Figura 8: Exemplos de armas criadas para o jogo *Galactic Arms Race* com a técnica descrita em (HASTINGS; GUHA; STANLEY, 2009b).

desenvolvido seja sorteado, garantindo assim um pouco de diversidade e boa qualidade dos itens gerados.

A forma como o jogador contribui para o processo evolutivo é feito de forma inconsciente, sem que isso afete sua imersão em relação ao ambiente criado pelo jogo. Porém, da mesma forma que (KERSSEMAKERS et al., 2012), embora o processo consiga evoluir armas incrivelmente diferentes, este se baseia fortemente na necessidade de um ator humano para a seleção de novos conteúdo. Além disso, o processo desenvolvido só é válido para utilização *online* e ativa, excluindo qualquer possibilidade de utilização em etapas de desenvolvimento.

Também utilizando uma combinação de SBPCG e EDPCG, (CARDAMONE; LOIACONO; LANZI, 2011) sugere um processo onde ferramentas separadas são utilizadas para a evolução de pistas para um jogo de corrida. Sua intenção é evoluir pistas que sejam interessantes aos jogadores, avaliando de forma coletiva as melhores pistas geradas pelo sistema. A proposta utiliza uma interface *web* onde jogadores avaliam de forma coletiva quais são as suas pistas favoritas. Um sistema evolutivo utiliza os dados obtidos pela plataforma web para a geração de novas pistas, que ficam então disponíveis dentro do jogo e também passam a ser avaliadas pelos jogadores através da interface *web*. Embora baseado em EDPCG, este processo pode ser visto como completamente *offline*, uma vez que tanto a avaliação como a geração de novos conteúdos é efetuada fora do jogo.

De acordo com os resultados apresentados, a metodologia conseguiu obter melhorias na satisfação dos usuários em relação ao conteúdo gerado. Um problema encontrado entretanto, foi a presença de pistas muito semelhantes a pistas já existentes sendo

adicionadas ao sistema. Com isso, as metodologias propostas em (CARDAMONE; LOIACONO; LANZI, 2011) não são cabíveis ao problema aqui apresentado; não sendo possível de ser aplicado de forma *online*, tendo a necessidade de atores humanos e constantemente gerando conteúdo similar.

De forma similar, (RISI et al., 2012) cria uma forma de EDPCG que utiliza redes sociais em conjunto com SBPCG para gerar uma forma de evolução colaborativa entre os jogadores. Sendo a evolução de novas flores parte principal da mecânica do jogo modelo utilizado, munido de um mercado virtual interno, esta técnica gera uma dinâmica de colaboração entre os jogadores em prol da procura de novos conteúdos diferenciados. Sementes são comercializadas entre os jogadores em um mercado global utilizando uma moeda virtual, como consequências de oferta e demanda do mercado, um grande incentivo em busca de novas flores é criado. A Figura 9 apresenta um exemplo de flores geradas com fractais através do processo colaborativo de jogadores.

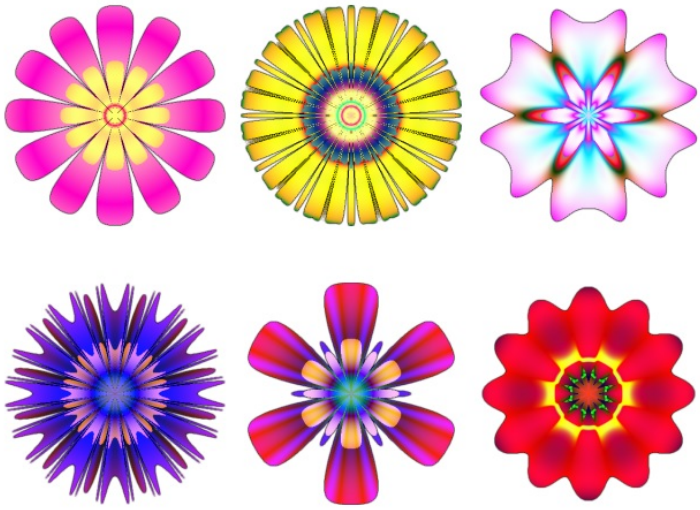


Figura 9: Exemplos de flores criadas com fractais através do processo colaborativo de jogadores em (RISI et al., 2012).

Mesmo sendo capaz de influenciar fortemente a evolução de conteúdo diversificado através de mecânicas de oferta e demanda, assim como o mercado real, a metodologia proposta depende fortemente de uma comunidade ampla para seu funcionamento. Com isso a aplicação de tal técnica se torna inviável para problemas onde a solução deve ser imediata, não sendo efetiva para o objetivo proposto nesta pesquisa.

O estudo em (LIAPIS; YANNAKAKIS; TOGELIUS, 2013b) evolui abstrações de mapas utilizados em jogos de estratégia em tempo real de sucesso como *Starcraft* (BLIZZARD ENTERTAINMENT, 1998). O método proposto é uma solução completa-

mente *offline*, que otimiza rascunhos de mapas gerados por humanos ou agentes artificiais através de um algoritmo de busca.

Nesta solução, mapas possuem um genótipo codificado que podem representar mapas não jogáveis. Devido a isso, um algoritmo genético com duas populações, uma com indivíduos viáveis e outra com indivíduos inviáveis, são mantidas e evoluídas separadamente. Indivíduos são atribuídos à devida população assim que gerados dependendo do seu estado: viável ou inviável. O estudo também analisa o uso de diferentes funções de avaliação e o uso de uma ou múltiplas funções como nota de avaliação final, demonstrando que o uso de várias funções limitam a eficiência da otimização, mas tendem a obter mapas mais apropriados para uso. A Figura 10 mostra exemplos de mapas evoluídos com o uso de diferentes funções de avaliação.

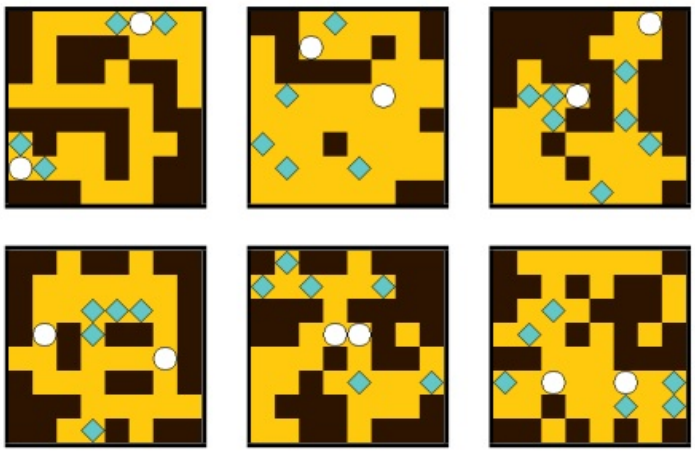


Figura 10: Exemplos de mapas evoluídos com o uso de diferentes funções de avaliação em (LIAPIS; YANNAKAKIS; TOGELIUS, 2013b).

Dando continuidade ao trabalho desenvolvido em (LIAPIS; YANNAKAKIS; TOGELIUS, 2013b), (LIAPIS et al., 2013) adiciona a interação humana ao processo de evolução. Atores humanos passam a controlar o peso das diversas dimensões da função de avaliação através de um sistema de classificação baseado em preferência pessoal. Os resultados mostram que a técnica introduzida fornece mais controle e converge conteúdos mais rapidamente em direção à real preferência dos usuários se comparado com outras técnicas previamente estudadas.

Embora não tenham foco em diversificação de conteúdo gerado, tais estudos mostram a importância da decisão da função de avaliação para a evolução de conteúdos como mapas para jogos. Diversos aspectos podem ser considerados, podendo ter grande influência nos resultados finais. Além disso, uma reflexão sobre conteúdos viáveis e inviá-

veis é aberta e pode ajudar no processo de decisão de representação genotípica e fenotípica dos itens gerados.

Em contraste aos algoritmos genéticos convencionais, o algoritmo de Busca Inovativa (*Novelty Search* - NS), introduzido e detalhado em (LEHMAN; STANLEY, 2008) e (LEHMAN; STANLEY, 2011a), vem sido utilizado em diversos problemas de busca por inovação. Embora tenha sido desenvolvido para auxílio em soluções de problemas combinatórios que confundem a solução, as características divergentes da NS beneficiam muito a busca por novos contextos.

A NS utiliza um cálculo de dispersão, normalmente baseado em uma função de distância comportamental, para classificar o nível de inovação dos indivíduos da população em cada geração. Indivíduos recebem notas baseadas em quão esparsos são as regiões em que se encontram; indivíduos em regiões muito esparsas recebem altas notas inovação, enquanto indivíduos em regiões densas recebem baixas notas de inovação. Um arquivo de indivíduos inovadores é mantido; sempre que um indivíduo recebe uma alta nota de inovação, ele é adicionado permanentemente ao arquivo, que por sua vez também é utilizado para o cálculo de dispersão em todas as gerações. Desta forma a NS mantém uma busca por inovação baseada no tipo de função de distância definido, o arquivo permanente de indivíduos inovadores ajuda a manter um histórico de onde a busca já esteve e incentiva novas regiões a serem exploradas.

Por ser uma busca divergente, o uso da NS pode auxiliar na busca por conteúdo diversificado, não ficando concentrada em regiões de máximo ou mínimo local como algoritmos genéticos convencionais.

O sucesso do uso da NS em problemas de otimização, entretanto, está fortemente conectado à definição da função de inovação. Além disso, (CUCCU; GOMEZ, 2011) mostra que em problemas onde o espaço de busca é muito grande, a NS sozinha não é capaz de superar os resultados obtidos pelo Algoritmo Genético convencional. (CUCCU; GOMEZ, 2011) demonstra também que uma combinação entre a busca por diversidade e busca por performance convencional pode melhorar muito os resultados obtidos dependendo do problema estudado.

A seguir são apresentados alguns exemplos onde a NS tem sido aplicada com sucesso para a evolução de conteúdo diversificado.

Extendendo o trabalho realizado em (LIAPIS; YANNAKAKIS; TOGELIUS, 2013b) e (LIAPIS et al., 2013), (LIAPIS; YANNAKAKIS; TOGELIUS, 2015) e (LIAPIS;

YANNAKAKIS; TOGELIUS, 2013a) adicionam o conceito de NS ao algoritmo de duas populações para a evolução de conteúdo viável em um jogo de estratégia em tempo real.

De forma similar, as soluções propostas que utilizam a NS também mantém duas populações separadas, uma com indivíduos viáveis e outra com indivíduos inviáveis. Estas possuem restrições de troca genética entre si e novos indivíduos são atribuídos às suas devidas populações com base em sua viabilidade. Duas soluções utilizando a NS são propostas: na primeira, a população inviável utiliza a NS para minimizar a distância entre seus indivíduos e os indivíduos da população viável, enquanto a população viável utiliza a NS para maximizar a inovação; na segunda proposta, a NS é utilizada para maximizar a inovação em ambas as populações.

Os resultados obtidos pelo autor mostram que a primeira configuração é capaz de convergir resultados satisfatórios mais rapidamente que os métodos previamente estudados, enquanto a segunda configuração é capaz de obter uma maior diversidade de soluções finais. Tais resultados são interessantes pois mostram que as características de busca por diversidade introduzidos pela NS podem ser utilizada de forma diferente da proposta originalmente e ainda obter bons resultados diversificados. A utilização de populações viáveis e inviáveis, contudo, é uma situação que pode ser evitada na elaboração da representação genotípica e fenotípica do problema.

Em (LIAPIS; YANNAKAKIS; TOGELIUS, 2013c) é introduzida uma técnica que combina procura pelo gradiente, através de *backpropagation*, e NS para criação de uma ferramenta de auxílio e melhoria de criatividade humana, oferecendo formas para se refinar rascunhos de terrenos gerados adicionando detalhes e dissimilaridades entre soluções.

A NS é utilizada para gerar redes neurais artificiais dissimilares, que são treinadas para se aproximarem de um rascunho apresentado, e então providenciar mapas com maiores níveis de detalhe. Em processos iterativos, os resultado são apresentados à um ator humano, que rejeita, aceita ou altera as soluções apresentadas, e então continuam a evoluir mais detalhes até formar mapas completos com alta resolução, podendo ou não divergir do rascunho inicial oferecido para o treinamento. A Figura 11 apresenta a interface utilizada pelo ator para escolher um mapa de maior resolução para prosseguir. A ferramenta dá suporte ao processo criativo do ator humano, oferecendo variações da proposta inicial apresentada, enquanto o mantém em controle sobre o conteúdo sendo evoluído durante todo o processo através das etapas iterativas. Os resultados obtidos mostram que o uso da NS é benéfico para a obtenção de conteúdos divergentes sem reduzir a velocidade

de refinamento iterativo dos mapas.

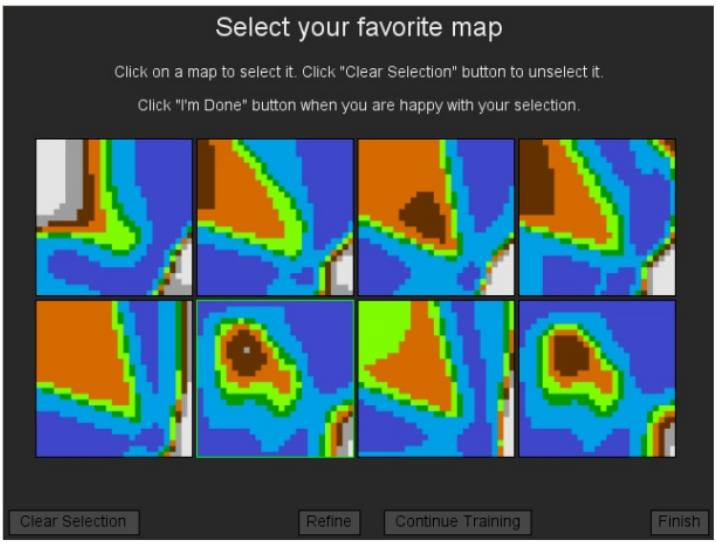


Figura 11: Tela de seleção da ferramenta implementada em (LIAPIS; YANNAKAKIS; TOGELIUS, 2013c) onde o ator escolhe um mapa de maior resolução para prosseguir.

Também utilizando uma combinação de processos com interatividade humana e NS, (WOOLLEY; STANLEY, 2014) evolui comportamentos de navegação para robôs em um labirinto complexo. A solução permite que um ator humano escolha não só os parâmetros de evolução entre cada iteração, como também permite que ele escolha qual método será usado para as próximas iterações e quais indivíduos devem ser evoluídos através da NS. A combinação da intuição humana em encontrar comportamentos que sejam capazes de solucionar o problema, com a capacidade de diversificar as opções sendo oferecidas quando desejado contribuem para uma convergência mais rápida em direção a solução. Os resultados obtidos mostram que a taxa de qualidade da evolução é acelerada e a fatiga do ator humano é reduzida significativamente.

As técnicas apresentadas tanto em (LIAPIS; YANNAKAKIS; TOGELIUS, 2013c) quanto em (WOOLLEY; STANLEY, 2014), são técnicas estritamente *offline* que necessitam da intuição de um ator humano para seu funcionamento. Avaliando estas e outras técnicas com base em interação humana previamente apresentadas, podemos concluir que qualquer solução baseada em EDPCG não é satisfatória para a resolução do problema aqui proposto. Quando dependente de um ator humano, a solução se torna dependente de contexto e normalmente é restrita a funcionamento somente *online* ou somente *offline*. Deseja-se uma solução que seja livre da necessidade de interação humana e que seja capaz de separar diferentes soluções de forma rápida e eficiente, garantindo a diversidade e qualidade dos itens gerados.

Neste sentido, a solução proposta em (LEHMAN; STANLEY, 2011b) se aproxima perfeitamente do objetivo geral desta pesquisa, sendo capaz de evoluir soluções distintas sem a necessidade de atores humanos. (LEHMAN; STANLEY, 2011b) sugere a combinação da função de avaliação convencional com a função de inovação proposta pela NS, através de um algoritmo com otimização baseado em eficiência à Pareto, para a evolução de criaturas virtuais com capacidade de locomoção. A Figura 12 apresenta um exemplo de criaturas virtuais com capacidade de locomoção razoável e diferentes morfologias encontradas em uma única execução do algoritmo.

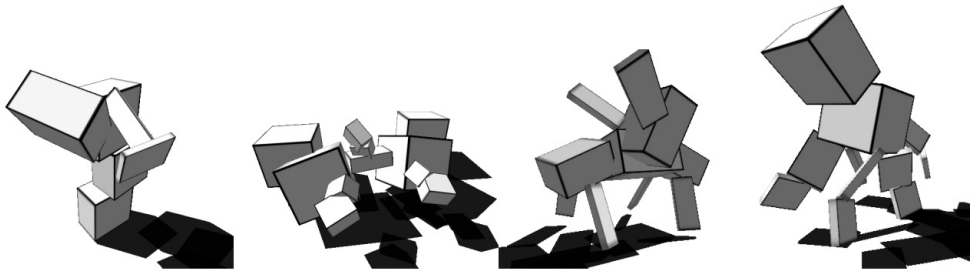


Figura 12: Exemplo de criaturas virtuais com capacidade de locomoção razoável e diferentes morfologias encontradas em uma única execução do algoritmo apresentado em (LEHMAN; STANLEY, 2011b).

O algoritmo utilizado em questão é o NSGA-II, um algoritmo evolutivo multiobjetivo com eficiência à Pareto proposto e detalhado em (DEB et al., 2002). Além da combinação entre performance e inovação, a solução propõe a utilização de uma métrica de Nota Local, que quando combinada à métrica de inovação em um algoritmo de eficiência à Pareto é capaz de manter nichos morfológicos durante toda a busca. Nichos morfológicos são regiões que possuem similaridades no espaço das morfologias, este gerado pela função de distância definida para o algoritmo de NS. Diferentes nichos morfológicos possuem diferentes capacidades funcionais, a solução utilizada se mostra capaz de explorar eficientemente cada nicho, obtendo os melhores indivíduos em cada um deles.

Embora utilizada para evoluir criaturas virtuais com capacidade de locomoção, se bem elaborada, a solução pode ser aplicada no contexto de evolução de mapas para jogos de exploração e aventura. A solução porém, explora todos os nichos morfológicos existentes, mantendo indivíduos de nichos tanto de alta como baixa capacidade.

O conceito de critério mínimo para a NS é introduzido em (LEHMAN; STANLEY, 2010) e depois elaborado de forma mais genérica em (GOMES; URBANO; CHRISTENSEN, 2012), sem a necessidade de parâmetros iniciais dependentes de contexto.

A ideia por trás do critério mínimo é que indivíduos precisam atingir uma capacidade funcional mínima aceitável para que sejam elegíveis para reprodução. Desta forma,

qualquer indivíduo da população que não tenha atingido um valor funcional mínimo em relação a função de avaliação, é atribuído a uma nota zero de inovação, sendo escolhidos para reprodução somente caso todos os outros indivíduos restantes tenham nota zero de inovação. Desta forma, a NS exclui, ou minimiza, a quantidade de indivíduos "ruins" durante o processo evolutivo, e a exploração, embora continue divergente, tende a navegar em áreas com melhores capacidades funcionais.

Os resultados obtidos mostram que, especialmente a versão de critério mínimo proposta em (GOMES; URBANO; CHRISTENSEN, 2012), é capaz de superar outras configurações utilizando somente Algoritmo Genético e NS, especialmente em problemas com o de labirintos de difícil navegação. O uso de um critério mínimo pode auxiliar na obtenção de indivíduos com melhor capacidade funcional ao mesmo tempo que permite que a NS busque por inovação.

A seguir são apresentados alguns trabalhos onde o principal contexto é a geração procedural de mapas, mesmo não havendo foco em diversidade nos resultados obtidos. São estudadas principalmente as formas de representação e avaliação de mapas, visando buscar inspiração e conceitos para aplicação nesta pesquisa.

Em (LINDEN; LOPES; BIDARRA, 2014) é apresentado um estudo específico sobre métodos de PCG para mapas de jogos de aventura. Ele resume práticas comuns, discute prós e contras de diferentes técnicas e identifica desafios promissores. O estudo descreve as técnicas atuais como boas em performance, mas deficientes em relação à controle e precisão sobre o conteúdo gerado.

Em geral, técnicas de PCG para criação de mapas consistem em três elementos: um modelo representativo, um método para montar o modelo representativo e um método para criar a geometria real do mapa com base em um modelo representativo qualquer. O estudo detalha métodos que são relevantes a técnicas de PCG e os classifica em autômatos celulares, gramáticas generativas, algoritmos genéticos e baseados em restrições. As análises sobre as técnicas de PCG baseadas em algoritmo genético revelam diversas formas interessantes de representação genotípica e fenotípica para os mapas. Também são estudadas diversas formas de gerar a função de avaliação, detalhando as vantagens e desvantagens de cada uma delas.

O estudo revela que várias técnicas presentes são capazes de gerar diversos tipos de representações finais e todas dependem do contexto e intenção do conteúdo gerado. As técnicas em geral são rápidas o suficiente, mesmo o contexto onde são aplicadas não requerendo uma resposta imediata, com os mapas sendo gerados durante telas de carre-

gamento ou em segundo plano enquanto o jogador já está em jogo para gerar os próximos níveis.

Em um estudo mais específico, (ASHLOCK; LEE; MCGUINNESS, 2011) propõe diferentes formas de representação de mapas e formas de avaliação do conteúdo gerado. Foco é dado para mapas que tenham uma estrutura parecida com a de um labirinto, tornando a exploração um dos principais focos do conteúdo gerado.

Quatro principais formas de representação são sugeridas e estudadas: direta, cromática, positiva e negativa. Em todas as representações o mapa é dividido em forma de grade, e as possíveis entradas em cada setor é aberto ou fechado, com aberto representando lugares onde se pode caminhar e fechado representando paredes ou obstáculos. Na representação direta os blocos abertos e fechados são representados diretamente em um único e longo gene; a representação cromática atribui cores à cada setor para controlar a movimentação de um agente que irá gerar toda a área aberta do mapa; a representação positiva possui em seu gene estruturas que são posicionadas em uma grade completamente aberta, formando diversas paredes e obstáculos; e a representação negativa inicia com um mapa completamente fechado e possui em seu genótipo as estruturas que serão abertas para gerar o mapa, como salas e corredores. A Figura 13 apresenta exemplos de mapas criados pela representações cromática, positiva e negativa. Além das representações e das funções de avaliação definidas, a técnica utiliza de um sistema de *checkpoints* para garantir a conectividade e comprimento dos caminhos dentro de um mapa.

Figura 13: Da esquerda para a direita, exemplos de mapas criados pelas representações cromática, positiva e negativa em (ASHLOCK; LEE; MCGUINNESS, 2011).

O estudo, além de contribuir com diferentes formas de representação e funções de avaliação, mostra que diferentes técnicas podem ser utilizadas com as mesmas funções de avaliação, mas resultam em mapas completamente diferentes. Isso revela que além da função de avaliação, a escolha das técnicas e representações dos mapas pode afetar de forma drástica os resultados obtidos.

Outro estudo interessante, (VALTCHANOV; BROWN, 2012), apresenta uma técnica para representação e evolução de mapas que possuem critérios mais complexos em sua estrutura. Os mapas gerados dão suporte a salas com eventos e regiões separadas por salas únicas, onde geralmente, em jogos de aventura, o jogador é forçado a obter uma chave ou completar algum quebra-cabeça para obter acesso.

O cromossomo utilizado neste método é uma estrutura de árvore onde cada nó representa uma peça a ser posicionada e a porta a ser utilizada para conectar aos seus pais. As peças são pré-definidas em relação a tamanho, forma e posicionamento das portas. Durante o processo de tradução, a árvore é processada iniciando da raiz (sala inicial) e adicionando as salas de seus nós filhos. Novas peças são posicionadas e rotacionadas de forma a se conectar às salas anteriores, caso não seja possível adicionar a nova peça sem gerar colisão entre peças já existentes, o nó que a representa e todos os seus descendentes são removidos do genótipo. A Figura 14 mostra exemplos de mapas gerados através desta técnica.

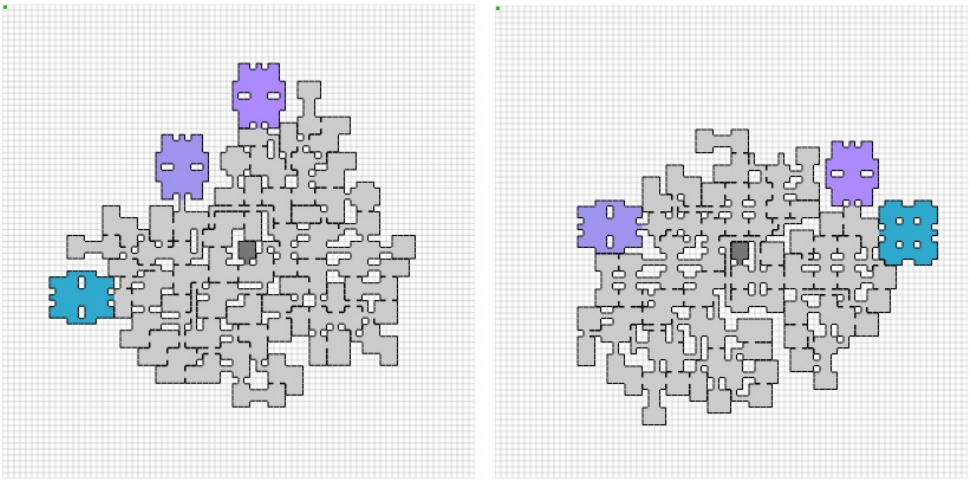


Figura 14: Exemplos de mapas gerados através da técnica proposta em (VALTCHANOV; BROWN, 2012).

Os mapas resultantes são visualmente interessantes e tendem a possuir boas características para um jogo de aventura e exploração. Porém, foi identificado que com a adição de novas restrições para separações de áreas e salas de evento, surge um grande problema de máximo local, com o algoritmo gerando constantemente soluções muito parecidas.

(CARDAMONE et al., 2011) estuda formas de utilizar a PCG para criar mapas interessantes para jogos de tiro em primeira pessoa. Em sua solução, são sugeridas quatro formas de representação de mapas, sendo duas baseadas nas formas apresentadas em

(ASHLOCK; LEE; MCGUINNESS, 2011): a positiva, adicionando paredes e obstáculos à um mapa inicialmente aberto; e a negativa, abrindo corredores e salas em um mapa inicialmente fechado. Das outras duas soluções, uma se baseia em um agente artificial para cavar regiões abertas em um mapa inicialmente fechado, e outra em uma estrutura onde o mapa é subdividido em grades maiores e paredes são removidas formando as conexões entre as áreas. A Figura 15 apresenta exemplos de mapas criados pelas diferentes representações propostas.

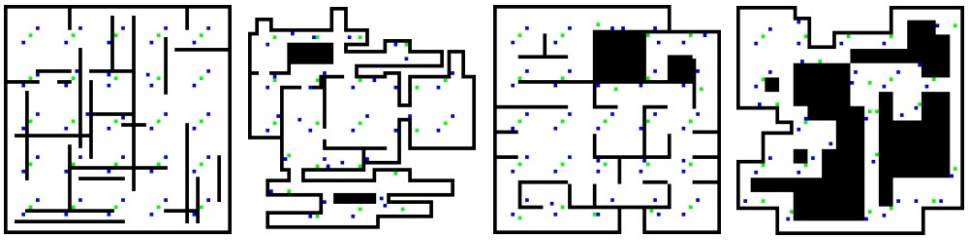


Figura 15: Exemplos de mapas criados pelas diferentes representações propostas em (CARDAMONE et al., 2011).

De acordo com o objetivo proposto em (CARDAMONE et al., 2011), as representações com maior potencial foram a positiva e a baseada em grade, pois oferecem maior tempo de combate entre jogadores no ambiente simulado de um jogo de tiro. Em contrapartida (LANZI; LOIACONO; STUCCHI, 2014) utiliza a representação negativa para criar mapas balanceados para jogos do mesmo estilo, alegando que os mapas gerados pela representação negativa possuem uma estrutura interessante, com vários corredores e grandes salões. A Figura 16 mostra exemplos de mapas gerados em (LANZI; LOIACONO; STUCCHI, 2014) utilizando somente a representação negativa.

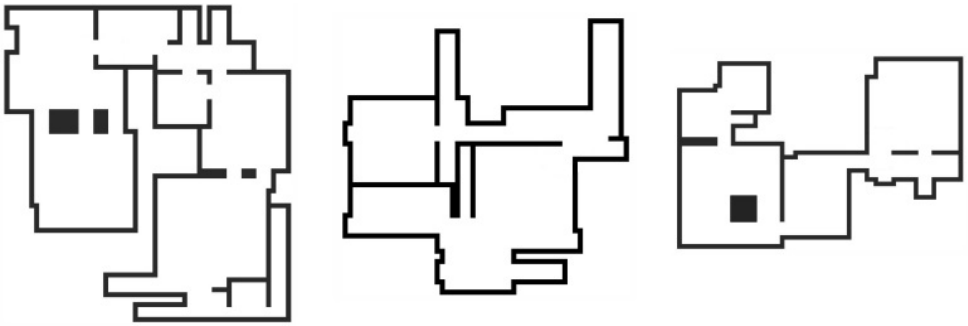


Figura 16: Exemplos de mapas gerados em (LANZI; LOIACONO; STUCCHI, 2014) que utilizam somente a representação negativa proposta em (CARDAMONE et al., 2011).

Se comparado com a representação negativa apresentada em (ASHLOCK; LEE; MCGUINNESS, 2011), a representação sugerida por (CARDAMONE et al., 2011), e utilizada por (LANZI; LOIACONO; STUCCHI, 2014), é mais simples e não necessita

do sistema de *checkpoints* para gerar mapas interessantes. Por isso, e por gerar estruturas interessantes, esta representação pode ser interessante para jogos de aventura e exploração.

1.4 Estrutura da Dissertação

Este documento está dividido em cinco capítulos: Introdução, Metodologia, Desenvolvimento, Experimentos e Resultados e Conclusão.

No capítulo Introdução é contextualizado o conceito de PCG, tanto de forma geral, quanto específica em relação à pesquisa, mostrando algumas de suas aplicações no mercado e em pesquisas. Também são apresentados os objetivos definidos e a justificativa da realização do trabalho.

Em Metodologia são detalhadas as principais técnicas e temas pesquisadas e utilizadas como base neste trabalho, abrangendo as áreas de Algoritmos Genéticos, eficiência à Pareto e separação de nichos.

Já no capítulo Desenvolvimento são descritas todas as modificações e técnicas utilizadas para a implementação da solução proposta para o problema.

Em Experimentos e Resultados são explicados os testes que foram executados para a validação do sistema implementado, detalhando os procedimentos seguidos e apresentando com gráficos e tabelas a análise dos comportamentos e valores obtidos.

Por fim, no capítulo Conclusão são considerados os resultados obtidos frente aos objetivos propostos, assim como a indicação de trabalhos futuros para a continuidade da pesquisa.

2 Metodologia

Neste capítulo são apresentadas diversas metodologias relacionadas às métricas de busca por inovação, expondo seus parâmetros de busca e fazendo uma breve análise em relação ao potencial de suas aplicações no problema proposto.

2.1 Novelty Search

Algoritmos evolutivos são algoritmos de otimização que convergem soluções em direção à uma função de avaliação global. Em contraste, a Busca Inovativa (*Novelty Search* - NS) (LEHMAN; STANLEY, 2008) (LEHMAN; STANLEY, 2011a) é uma técnica de evolução divergente baseada na característica de diversificação da evolução natural, a qual é capaz de criar e manter várias espécies que dão suporte à vida. Desta forma a NS recompensa comportamentos inovadores ao invés do progresso em direção à um objetivo fixo, explorando eficientemente o espaço de busca e eventualmente encontrando indivíduos que solucionem o problema, mesmo sem promover um gradiente rumo à nota de avaliação. De fato, a NS vem sendo aplicada de forma eficiente e bem sucedida em diversos problemas (LEHMAN; STANLEY, 2011b) (LIAPIS; YANNAKAKIS; TOGELIUS, 2013c) (WOOLLEY; STANLEY, 2014).

A implementação da NS requer quase nenhuma modificação em qualquer algoritmo evolutivo exceto pela troca da função de avaliação objetiva por uma métrica de inovação, esta tem por sua vez a função de medir o quão diferente um indivíduo é em relação a outros. A ideia principal é que ao invés de recompensar performances em direção a um objetivo, a NS recompensa a diversidade em relação a comportamentos anteriores. As Figuras 17 e 18 apresentam as modificações básicas necessárias para a implementação da NS.

A inovação de um indivíduo recentemente gerado é computada em relação a comportamentos de um arquivo de indivíduos previamente inovadores e à população atual.

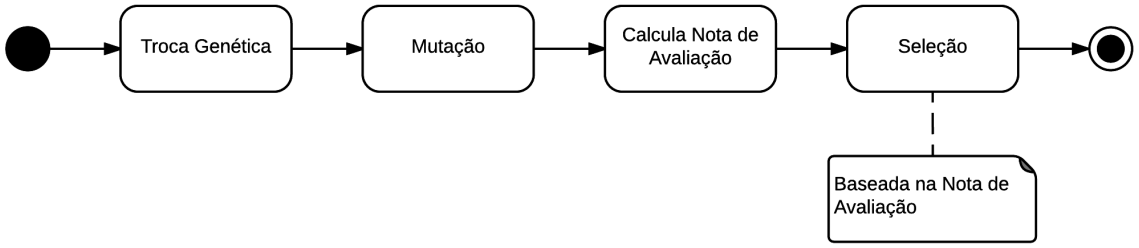


Figura 17: Diagrama de Atividades de um típico Algoritmo Genético.

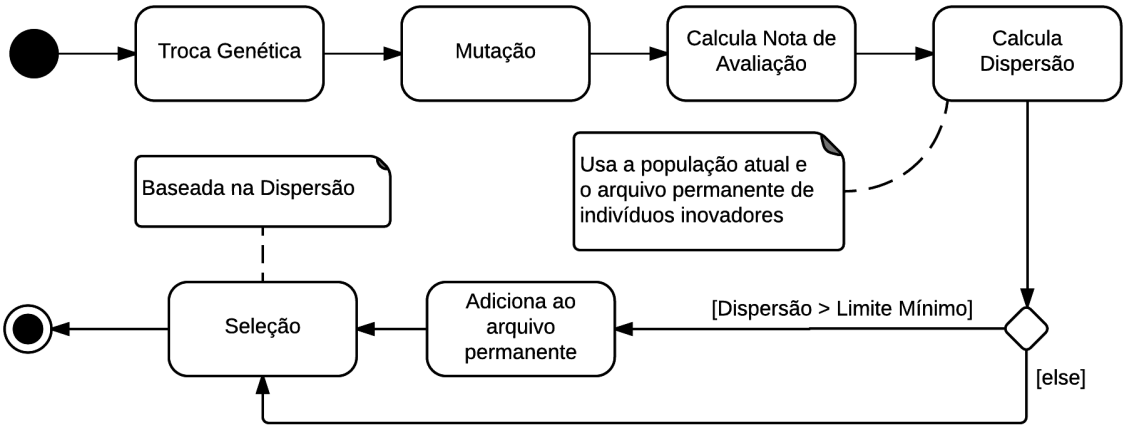


Figura 18: Diagrama de Atividades de um Algoritmo Genético de Busca Inovativa.

O arquivo, inicialmente vazio, recebe novos indivíduos quando estes são significativamente diferentes de indivíduos anteriores, i.e., se possuem um valor de inovação acima de um limite computado dinamicamente ou pré-definido.

A métrica de inovação caracteriza o quão distante um indivíduo está em relação ao resto da população atual e seus predecessores no espaço comportamental, baseando-se na dispersão de qualquer ponto neste espaço comportamental. Áreas com altos índices de aglomeração de pontos visitados são menos inovadoras, e portanto menos recompensadas.

Um simples cálculo de dispersão em um certo ponto é a média das distâncias entre este ponto e os k-vizinhos mais próximos à ele, onde k é uma constante fixa definida empiricamente. Intuitivamente, se a distância média até os vizinhos mais próximos de um ponto for grande, esta é uma área esparsa; caso seja um valor pequeno, o mesmo está localizado em uma área densa; a Figura 19 representa os pontos P1 e P2 em regiões mais

e menos densas respectivamente. A dispersão ρ em um ponto x é dada pela equação 2.1

$$\rho(x) = \frac{1}{k}\sum_{i=0}^{k} dist(x, \mu_i), \tag{2.1}$$

onde μ_i é o i-ésimo vizinho mais próximo de x com respeito a métrica de distância $dist$, que é uma medida de diferença comportamental dependente de domínio entre dois indivíduos no espaço de busca. O cálculo de vizinhos mais próximos deve levar em consideração tanto indivíduos da população atual como do arquivo de indivíduos previamente inovadores. Candidatos em regiões mais esparsas dentro deste espaço de busca comportamental recebem assim maiores valores de inovação, guiando, desta forma, a busca em direção ao que é novo, sem objetivos explícitos.

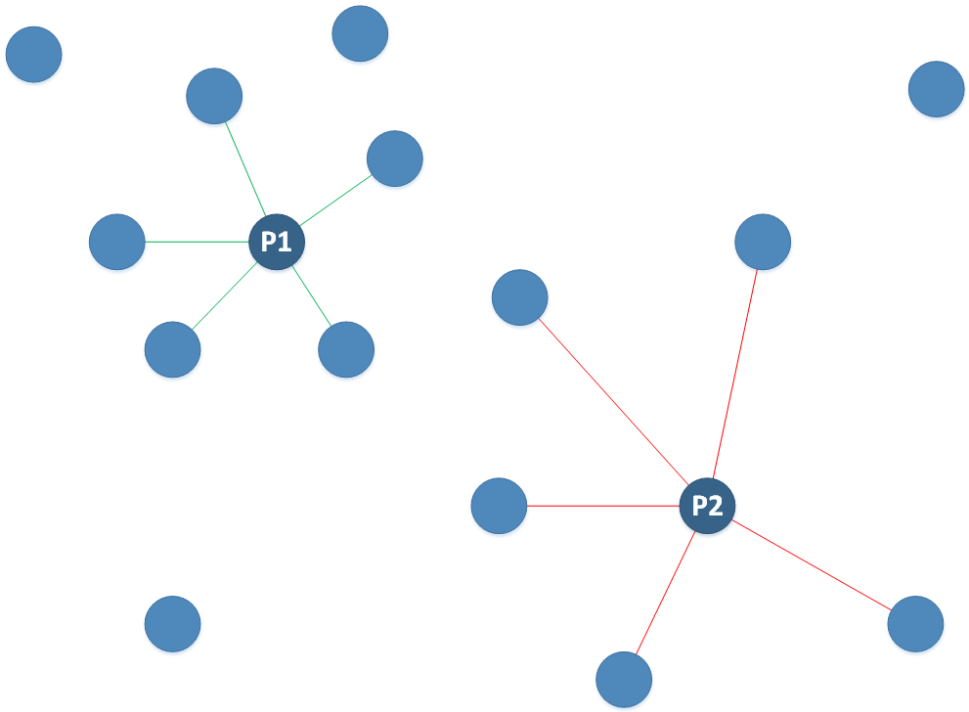


Figura 19: P1 está em uma região mais densa que P2.

A geração atual, somada ao arquivo de indivíduos inovadores, fornece uma amostra completa de onde a busca esteve e onde está atualmente; sendo assim, ao tentar maximizar a métrica de inovação, o gradiente de busca se direciona simplesmente ao que é novo, sem qualquer objetivo explícito.

Após substituir a função de avaliação objetiva pela métrica de inovação, o algoritmo evolutivo continua a operar normalmente, selecionando os indivíduos mais inovadores para reprodução. Ao longo das gerações, a população se espalha através do espaço de possíveis comportamentos, algumas vezes encontrando um indivíduo que solucione um

dado problema mesmo sem recompensar o progresso em direção à solução do mesmo.

Embora a nota de avaliação não influencie no gradiente de busca na NS, normalmente uma função de avaliação deve ser especificada para identificar os melhores parâmetros de controle para a busca, assim como auxiliar com o critério de parada ao encontrar um indivíduo que satisfaça tal função.

Ao invés de recompensar comportamentos inovadores como na maioria dos experimentos realizados com a NS, semelhante a forma utilizada em (LEHMAN; STANLEY, 2011b), que analisa o espaço de morfologias de criaturas virtuais, uma nova proposta de explorar o espaço de características fenotípicas, com a finalidade de recompensar características inovadoras que se diferenciam de outras já encontradas, pode melhorar ainda mais os resultados desta metodologia no âmbito desta pesquisa.

2.1.1 Minimal Criteria Novelty Search

A Busca Inovativa com Critério Mínimo (*Minimal Criteria Novelty Search* - MCNS) (LEHMAN; STANLEY, 2011b) é uma extensão da NS que leva em consideração outra característica importante da evolução natural: a função crítica para a evolução de qualquer organismo é a habilidade de sobreviver até ser capaz de se reproduzir.

Para cobrir esta característica, a MCNS implementa um critério mínimo em relação à função de avaliação global, indivíduos que satisfazem tal critério recebem o valor de inovação normalmente, enquanto indivíduos que não satisfazem o critério mínimo recebem o valor zero de inovação e são considerados para a reprodução somente caso nenhum outro indivíduo da população tenha atingido o critério mínimo.

Desta forma o valor de inovação de um indivíduo i na etapa de seleção para reprodução é dada pela equação 2.2

$$\text{inovação}(i) = \begin{cases} disp_i & \text{se } nota_i \geq cm \\ 0 & \text{caso contrário} \end{cases}, \tag{2.2}$$

onde $disp_i$ é o valor de dispersão calculado para indivíduo de acordo com a equação 2.1, $nota_i$ sua nota em relação a função de avaliação global e cm o critério mínimo predefinido.

Como a nota de avaliação normalmente já é calculada durante a NS, a implementação da MCNS não requer nenhuma tarefa específica nem qualquer medida adicional exceto a filtragem dos indivíduos baseados no valor já obtido.

Após o procedimento de avaliação de critério mínimo a seleção segue normalmente como na NS, selecionando indivíduos com base em suas notas de inovação. A operação de arquivo de indivíduos previamente inovadores não é alterada e funciona da mesma forma que na NS. Mesmo indivíduos que não atingem o critério mínimo são adicionados ao arquivo caso seus valores de inovação sejam bons o suficiente. Isso garante que a busca continue em direção à novos comportamentos sem ficar presos em áreas onde indivíduos não atingem o critério mínimo. A Figura 20 apresenta o diagrama de atividades da MCNS.

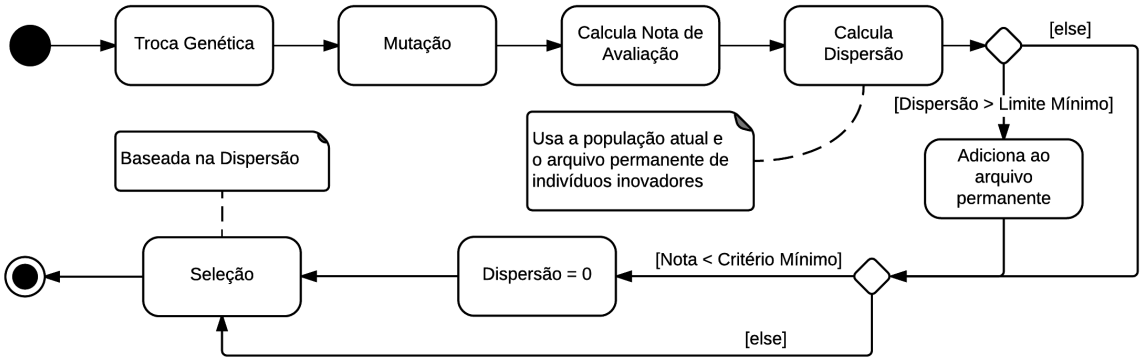


Figura 20: Diagrama de Atividades do Algoritmo Genético de Busca Inovativa com Critério Mínimo.

Como na MCNS os indivíduos que não atingem o critério mínimo são considerados falhas, até que um indivíduo que satisfaça o critério seja encontrado a busca é efetivamente aleatória. Desta forma, se um indivíduo que cumpra o critério não for facilmente encontrado na população inicial pode ser necessário fornecer um genoma previamente evoluído que satisfaça este critério.

2.1.2 Progressive Minimal Criteria Novelty Search

A Busca Inovativa com Critério Mínimo Progresso (*Progressive Minimal Criteria Novelty Search* - PMCNS) (GOMES; URBANO; CHRISTENSEN, 2012) é uma extensão da MCNS que visa aproveitar as restrições do espaço de busca comportamental gerado pela mesma, mas sem a necessidade de predefinir um critério mínimo dependente de domínio. Na PMCNS o limite para o critério mínimo é definido dinamicamente, e assim como na MCNS, somente indivíduos que possuam uma nota de avaliação maior que este limite são considerados para reprodução com seus valores originais de inovação, enquanto indivíduos que não atingirem o critério recebem o valor zero de inovação. Assim como na MCNS, a operação de arquivo de indivíduos previamente inovadores não é alterada.

O critério mínimo é iniciado com um valor mínimo teórico (cm_0) relacionado a função de avaliação (tipicamente zero), desta forma todos os indivíduos iniciais atingem o critério. Em cada nova geração o novo critério mínimo é obtido, inicialmente ordenando-se os indivíduos de forma crescente em relação a suas notas de avaliação e em seguida calculando a posição n através da equação 2.3

$$n = \frac{P}{100} \times N + \frac{1}{2} \ , \tag{2.3}$$

arredondado para o inteiro mais próximo, onde P $(0 \leq P \leq 100)$ é a porcentagem predefinida de indivíduos que devem ficar abaixo do critério mínimo, e N o tamanho da população. O valor v_n é então obtido tomando-se a nota de avaliação da posição n dos valores ordenados.

A fim de restringir progressivamente o espaço de busca, evitando consumir muito tempo com comportamentos com baixas notas de avaliação, o critério mínimo é estritamente crescente e suavemente incrementado com base no valor utilizado na geração anterior de acordo com a equação 2.4

$$cm_g = cm_{g-1} + max(0, (v_n - cm_{g-1}) \times S), \tag{2.4}$$

onde cm é o critério mínimo e S $(0 \leq S \leq 1)$ o parâmetro de suavização.

O valor de inovação de um indivíduo i na etapa de seleção para reprodução é dada pela equação 2.5

$$\text{inovação}(i) = \begin{cases} disp_i & \text{se } nota_i \geq cm_g \\ 0 & \text{caso contrário} \end{cases} \ , \tag{2.5}$$

onde $disp_i$ é o valor de dispersão calculado para indivíduo de acordo com a equação 2.1, $nota_i$ sua nota em relação a função de avaliação global e cm_g o critério mínimo calculado para a geração atual.

O parâmetro P controla a exigência do critério mínimo: 0 permite que todos indivíduos satisfaçam o critério (assemelhando-se à NS), enquanto 100 restringe o critério mínimo somente ao indivíduo com maior nota. O parâmetro S, por sua vez, controla a velocidade de adaptação do critério mínimo: com 0 o critério mínimo não sofre alterações ao longo das gerações, enquanto que com 1 o valor calculado na geração anterior é completamente ignorado.

Com a utilização desta técnica é possível obter os mesmos resultados da NS ou MCNS controlando o parâmetro S e o valor mínimo teórico inicial cm_0. Com $S = 0$ e $cm_0 = 0$ temos os mesmos resultados da NS, onde todos indivíduos serão sempre aceitos. Alterando cm_0 e mantendo $S = 0$, obtém-se os mesmos resultados da MCNS, onde cm_0 não sofrerá alterações ao longo do processo evolutivo e continuará a restringir os indivíduos da mesma forma. Assim sendo, o uso do PMCNS é interessante devido à sua facilidade para comparação entre combinações e técnicas diferentes, além da ausência da necessidade de definição de um parâmetro pré estabelecido dependente de domínio.

2.2 NSGA-II

Algoritmos Evolutivos Multiobjetivo (*Multiobjetive Evolutionary Algorithms* - MOEAs) são uma classe de algoritmos evolutivos que visam resolver problemas de otimização que possuem mais de uma função de avaliação. A presença de vários objetivos em um problema gera um conjunto de soluções ótimas ao invés de uma solução única, este conjunto de soluções é amplamente conhecido como soluções Pareto ótimas. Um bom MOEA deve encontrar o maior número de soluções que sejam o mais próximas das soluções Pareto ótimas possíveis, ao mesmo tempo que preserva a diversidade entre as soluções de uma mesma frente não dominada (nichos de soluções).

Soluções Pareto ótimas são aquelas que não são dominadas por nenhuma outra solução, ou seja, sua nota de avaliação para todos os objetivos é sempre maior ou igual às notas de avaliação de quaisquer outras soluções do conjunto. Desta forma, a não ser que mais informações sobre os objetivos sejam fornecidas, não é possível classificar dentre as soluções Pareto ótimas uma que seja considerada a melhor. Por este motivo a diversificação das soluções encontradas em uma mesma frente não dominada é desejável, representando assim as mais variadas possibilidades dentre as soluções encontradas. A Figura 21 apresenta os indivíduos da frente Pareto ótima e de uma segunda frente não dominada em um problema que possui dois objetivos distintos: A e B.

O Algoritmo Genético com Ordenação Não-dominante II (*Nondominated Sorting Genetic Algorithm II* - NSGA-II) é um MOEA proposto em (DEB et al., 2002), que oferece uma boa distribuição de soluções e uma convergência próxima da fronteira Pareto ótima verdadeira, calculados de forma rápida e eficiente.

O NSGA-II implementa uma ordenação de soluções não dominada, onde cada solução é associada a um conjunto de soluções por ela dominadas e a um contador de

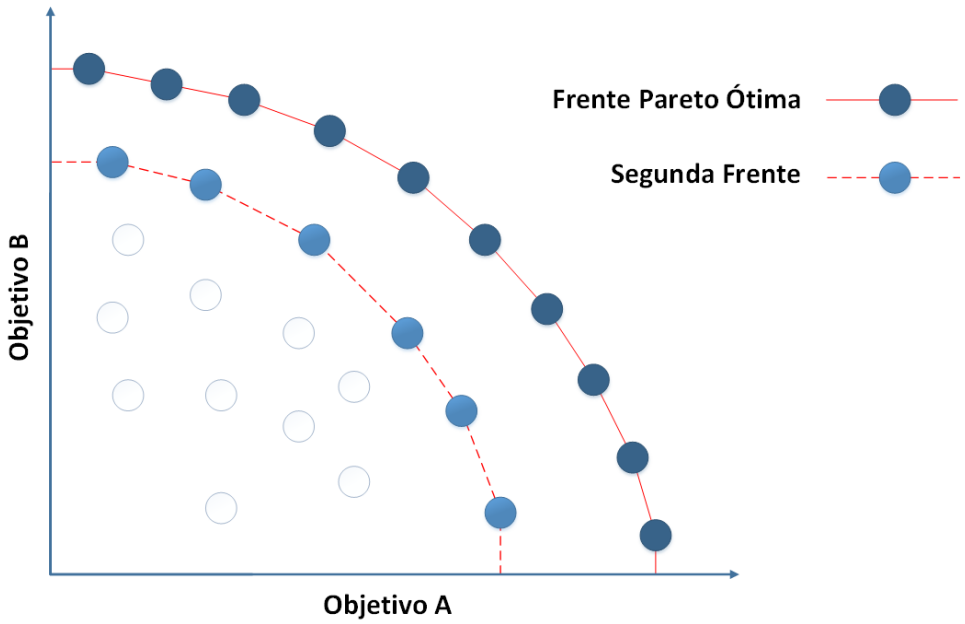


Figura 21: Frentes não dominadas em um problema multiobjetivo.

soluções que a dominam. As soluções são classificadas de acordo com a frente não dominada em que se encontram. Desta forma, a primeira frente é formada por soluções que não são dominadas por nenhuma outra solução da população; a segunda frente, por sua vez, é constituída por soluções que são dominadas por soluções da primeira frente e que não são dominadas por nenhuma das outras soluções restantes, e assim sucessivamente.

A fim de manter uma boa diversidade e distribuição das soluções dentro das frente não dominadas, o NSGA-II implementa um *operador de comparação de distribuição* (*Crowded-Comparision Operator*) para soluções em frentes não dominadas que não requer parametrização. Essa distribuição é estimada pelo *operador de distância de aglomeração* (*Crowding Distance*), que visa recompensar soluções que estejam em áreas menos densas da frente ao qual pertencem. Soluções dentro de uma mesma frente são classificadas com base na distância média até os pontos vizinhos em ambos os lados e para cada objetivo, calculada pelo perímetro médio do cuboide que envolve uma solução e cujos vértices são os pontos vizinhos na fronteira como mostra a Figura 22.

Assumindo que todo indivíduo i da população possui dois atributos:

1. Classificação não dominada ($ranque_i$);
2. Distância de aglomeração ($dist_i$).

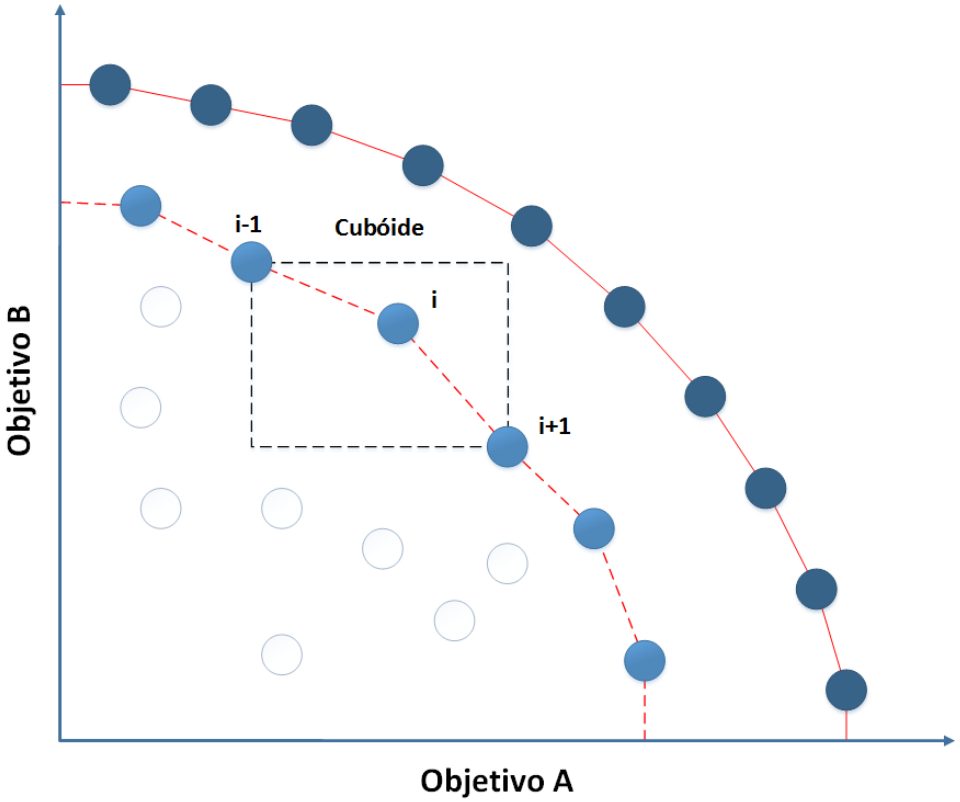


Figura 22: Representação do cuboide utilizado para o cálculo de distância de aglomeração.

O *operador de comparação de distribuição* ($\prec_n$) é definido pela equação 2.6

$$i \prec_n j \qquad \begin{array}{l} \text{se } (ranque_i < ranque_j) \\ \text{ou } ((ranque_i = ranque_j) \text{ e } (dist_i > dist_j)). \end{array} \tag{2.6}$$

Isto é, entre duas soluções com classificação não dominadas diferentes, escolhe-se aquele com menor (melhor) classificação. Caso contrário, se ambas as soluções pertencerem à mesma frente, então escolhe-se a solução localizada em uma região menos aglomerada.

Para gerar uma nova população P_{t+1} de tamanho N a partir de uma geração t qualquer, o NSGA-II combina as populações de progenitores (P_t) e descendentes (Q_t), os ordena com base no *operador de comparação de distribuição* $\prec_n$ e seleciona os N primeiros indivíduos. Com este procedimento o elitismo é garantido, já que indivíduos da população atual são sempre comparados com indivíduos previamente classificados como sendo as melhores soluções não dominadas até então. A nova população P_{t+1} é então utilizada para seleção, mutação e troca genética na criação da nova população de descendentes Q_{t+1}.

A escolha do NSGA-II como MOEA é interessante, pois além de sua velocidade e eficiência, fornece uma função de *distância de aglomeração* ($dist_i$). Esta, semelhante da forma como foi utilizada por (LEHMAN; STANLEY, 2011b), pode ser substituída por uma função de distância morfológica, utilizando o valor de dispersão ($disp_i$) para ordenar uma frente não dominada, sendo capaz de direcionar a diversidade durante a busca para o contexto morfológico em questão.

2.3 Diversificação Morfológica

Algoritmos evolutivos convergem toda a sua população em direção à função de avaliação global, normalmente isso implica em uma população final com indivíduos extremamente semelhantes em algumas de suas características fenotípicas. Esse comportamento ocorre por consequência da natureza do algoritmo evolutivo, que recompensa fortemente nichos onde a nota da função de avaliação é boa. As estratégias a seguir combatem este comportamento de convergência fenotípica e evoluem uma diversidade de criaturas virtuais com somente uma busca.

2.3.1 Balanceando Performance e Inovação

A estratégia utilizada em (LEHMAN; STANLEY, 2011b) emprega as características de exploração por inovação da NS para buscar novos nichos de indivíduos, definindo uma métrica para quantificar a diferença morfológica entre novos indivíduos e indivíduos anteriores.

Com uma métrica de distância puramente morfológica, porém, não é garantido que os indivíduos evoluídos somente com a NS sejam funcionais, ou seja, recompensar somente diferenciação morfológica não visa a melhoria em prol da função de avaliação, garantindo somente a diversidade das características avaliadas em questão e nada mais. Desta forma, uma extensão ao método de NS é proposta, visando contornar a falta de busca objetiva e balanceando duas frentes concorrentes (inovação e performance) com a utilização de um MOEA baseado em eficiência à Pareto. Esta estratégia pode ser vista como uma busca objetiva que encoraja inovação a fim de manter uma diversidade morfológica na população.

Entretanto, a simples combinação de uma função de avaliação global e inovação morfológica em um MOEA não dá suporte ao fato de que: diferentes nichos suportam diferentes níveis de nota de avaliação. Indivíduos em um MOEA baseado em eficiência à

Pareto são recompensados com base nas frentes não dominadas. Com isso, em uma frente não dominada com a inovação e a performance como dois objetivos separados, teremos em um extremo indivíduos que maximizam a inovação à custa de performance, enquanto em outro extremo temos indivíduos que maximizam a performance à custa de inovação, e entre ambos os extremos haverão várias combinações e trocas de valores entre performance e inovação. Desta forma, um indivíduo que tenha uma excelente nota de avaliação dentro de seu nicho, mesmo com a nota de avaliação medíocre em relação ao total da população, não será recompensado da forma como deveria.

2.3.2 Competição Local

A solução proposta em (LEHMAN; STANLEY, 2011b) para solucionar o problema de diferentes capacidades dos nichos é: limitar a competição entre indivíduos localmente em relação a seus nichos; ou seja, indivíduos competem entre si somente com outros indivíduos morfologicamente próximos. A ideia é explorar a capacidade dentro de cada nicho ao invés de explorar gananciosamente somente os melhores nichos.

A principal mudança é que a competição local dentro de um nicho transforma a nota de avaliação, que passa de uma medida global para uma medida relativa somente aos vizinhos próximos em seu nicho. Desta forma, o gradiente da busca passa de um balanço entre diversidade e performance global, que beneficia somente alguns nichos, para um balanço entre diversidade e performance dentro de um nicho local.

Na prática, transformar a função de avaliação global em local requer a comparação das performances de um indivíduo em relação a seus vizinhos. Quanto mais vizinhos forem superados maior será sua nota de avaliação local. Assim sendo, a implementação da competição local como uma extensão da NS é simples e direta, uma vez que a NS já calcula os vizinhos mais próximos de cada indivíduo. Durante o cálculo de dispersão, obtido pela média das distâncias entre os vizinhos mais próximos, o número de vizinhos com nota de avaliação menor que o indivíduo em questão é contado. Este número é então atribuído à nota de competição local deste indivíduo, representando sua performance relativa ao seu nicho. A Figura 23 apresenta o diagrama de atividades com as modificações necessárias na NS para a implementação da estratégia de competição local.

O valor de competição local é então combinado com o valor de dispersão em um MOEA. Esta nova estratégia pode ser vista como uma busca objetiva limitada a nichos com encorajamento a inovação em busca de novos nichos. Graças às características da eficiência à Pareto, indivíduos com boa nota de avaliação dentro de seus nichos tendem

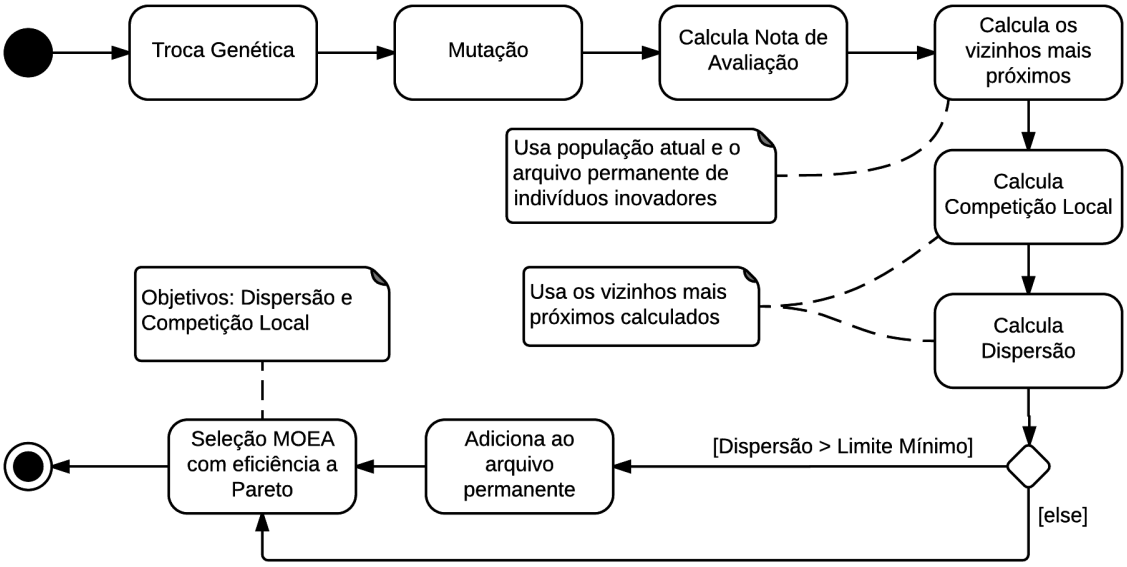


Figura 23: Diagrama de Atividades para o cálculo de Competição Local em um sistema de Busca Inovativa.

a ser mantidos ao mesmo tempo que indivíduos com altas notas de inovação, motivando assim a exploração e eventualmente encontrando novos nichos a serem otimizados.

Esta estratégia portanto se alinha perfeitamente com os objetivos propostos desta pesquisa: buscar as melhores e mais variadas soluções possíveis para um problema de forma rápida e eficiente. Separando o espaço de busca em nichos garante-se que os indivíduos encontrados serão os mais eficientes dentro de seus respectivos nichos. Com a função de distância é possível controlar quais serão as características diferenciais destes nichos, tipicamente características fenotípicas nas quais desejamos diversificação; no caso de (LEHMAN; STANLEY, 2011b) esta característica fenotípica é a morfologia das criaturas virtuais obtidas, e na aplicação escolhida para esta pesquisa são as características visuais e estruturais dos mapas gerados automaticamente.

3 Desenvolvimento

A seguir serão abordados os conceitos apresentados previamente a fim de dividir a população em nichos distintos e buscar os melhores indivíduos em cada um deles, explorando bem o espaço de busca e evitando a concentração em somente um máximo ou mínimo local. A estratégia apresentada a seguir tenta resolver o problema evoluindo somente uma população, sem a necessidade de nenhum controle direto sobre a separação dos nichos ou evolução do problema em questão a parte das funções de avaliação e distância.

Esta estratégia é desejável em contextos onde se faz necessária a geração de diversas soluções úteis que não sejam muito semelhantes entre si; contexto muito comum, por exemplo, em jogos com conteúdo gerado proceduralmente, que devem gerar artigos diversificados a fim de evitar repetição e monotonia; comum também em problemas onde a função de avaliação é uma estimativa da solução real, não sendo possível evoluir uma solução direta para o problema, desta forma podemos evoluir soluções variadas que se adaptam bem à função de avaliação e ofereçam uma base para o auxílio à tomada de decisão.

3.1 Estratégia de Competição Local

A combinação da NS com a estratégia de competição local em um MOEA com eficiência à Pareto, como apresentada em (LEHMAN; STANLEY, 2011b), provou ser eficiente na busca de diferentes morfologias em um problema de evolução de criaturas virtuais. Isso se deve ao fato da estratégia explorar bem o espaço de busca, identificando nichos com morfologias variadas e mantendo indivíduos com as melhores pontuações dentro de cada nicho. Uma grande vantagem desta estratégia é a capacidade de gerar indivíduos diversificados em uma única execução, enquanto que na maioria dos sistemas de busca evolutiva, descobrir tal diversidade se torna computacionalmente caro; embora execuções diferentes possam resultar em soluções variadas, uma única execução irá tipicamente convergir para

somente uma morfologia; além disso, não há garantia de que duas ou mais execuções irão resultar em morfologias diferentes por não haver nenhuma métrica de controle para a diversidade.

Nesta estratégia entretanto, não há nenhuma consideração real em relação à nota de avaliação do indivíduo, sendo utilizada somente a sua nota de competição local e seu valor de dispersão para seleção. Isso implica que nichos que possuem baixas capacidades de performance são mantidos durante a busca, competindo diretamente com indivíduos em nichos com capacidades superiores. No contexto de exploração de nichos esta estratégia é boa, pois mantém os melhores indivíduos de todos os possíveis nichos encontrados; em um contexto onde deseja-se encontrar somente os melhores nichos, contudo, estes indivíduos apenas ocupam espaço na população durante a busca, sem contribuir diretamente para a busca por novos nichos.

3.2 Critério Mínimo Global

A proposta de melhoria oferecida neste trabalho é, portanto, eliminar nichos que possuam uma capacidade funcional muito baixa, evitando que indivíduos destes nichos sejam mantidos por suas notas de competição local. Enquanto possa parecer trivial apenas adicionar a nota de avaliação global como um terceiro objetivo no MOEA, as implicações dessa alteração certamente influenciariam o gradiente de busca, afetando desta forma a procura por novos nichos. O objetivo da melhoria é somente eliminar nichos com notas de avaliação muito baixas, sem influenciar a forma como a busca se espalha pelo espaço das morfologias.

Para endereçar este problema de nichos com baixa capacidade funcional, considerou-se a premissa já elaborada em (LEHMAN; STANLEY, 2010) para a MCNS, de que a função crítica para a evolução de qualquer organismo é a habilidade de sobreviver até ser capaz de se reproduzir. Desta forma, esta pesquisa sugere a utilização de um critério mínimo relacionado à função de avaliação global durante o processo de seleção.

Para a introdução deste critério mínimo a implementação escolhida como base foi a PMCNS, pois não necessita da definição de um parâmetro para critério mínimo dependente de domínio e por ser capaz de simular a NS e a MCNS com simples alterações em seus parâmetros de controle. Porém, na implementação aqui proposta, a PMCNS utiliza o critério mínimo de forma diferente, atribuindo zero para a nota de competição local de indivíduos que não atingem o critério mínimo em relação a nota de avaliação

global, nunca alterando o valor de inovação calculado. Com isto, devido às características do MOEA, a busca continua a explorar os diversos espaços de morfologias, selecionando e mantendo em sua população somente indivíduos de nichos que atinjam o critério mínimo.

3.3 Estratégia de Seleção

Como MOEA para seleção, o algoritmo escolhido para esta pesquisa foi o NSGA-II. Além de rápido e eficiente este também foi o algoritmo utilizado em (LEHMAN; STANLEY, 2011b), onde teve seu *operador de distância de aglomeração* ($dist_i$) alterado em prol da diversidade morfológica. A escolha deste algoritmo não só permite uma melhor comparação com os estudos realizados, como também possui este *operador de distância de aglomeração* que pode ser utilizado em favor da busca por diversidade.

O *operador de distância de aglomeração* original ordena os indivíduos dentro de uma frente não dominada de forma que os indivíduos em áreas menos densas tenham maior nota, motivando assim a diversidade dentro de uma mesma frente. No caso onde temos como objetivos a competição local e a diversidade, isso significa que nenhum dos dois objetivos possui vantagem durante a seleção dentro de uma mesma frente somente com base em suas notas. O algoritmo segue escolhendo indivíduos de forma distribuída: tanto indivíduos com alto valor de competição global e baixo valor de diversidade; quanto indivíduos com alto valor de diversidade e baixo valor global; ou qualquer outro indivíduos com trocas de valores equilibradas dentro desta frente possuem chances de ser escolhidas contanto que estejam em áreas menos densas.

A modificação proposta em (LEHMAN; STANLEY, 2011b), que substitui o *operador de distância de aglomeração*, descrito pela equação 2.6, por um operador para verificar a diversidade morfológica; passa a ordenar os indivíduos em uma mesma frente de forma a dar preferência a indivíduos com alta diversidade morfológica. Nesta pesquisa, o espaço de busca de morfologias é resultante da função distância definida para o problema; sendo assim, utilizou-se o valor de dispersão ($disp_i$), calculado conforme a equação 2.1, para ordenar os indivíduos dentro de uma mesma frente não dominada, garantindo assim o favorecimento da diversidade morfológica dentro de uma mesma frente não dominada.

Assim sendo, o *novo operador de comparação de distribuição* ($\prec_{n_2}$) é definido

pela equação 3.1

$$\begin{aligned} i \prec_{n_2} j \qquad & \text{se } (ranque_i < ranque_j) \\ & \text{ou } ((ranque_i = ranque_j) \text{ e } (disp_i > disp_j)). \end{aligned} \tag{3.1}$$

Desta forma o *novo operador de comparação de distribuição* ($\prec_{n_2}$) continua a ordenar indivíduos com base em sua classificação não dominada, mas quando ambas as soluções pertencem à mesma frente, escolhe-se a solução localizada em uma região menos aglomerada em relação ao espaço das morfologias, e não em relação ao espaço dos objetivos. A Figura 24 é uma representação dos indivíduos selecionados com base na metodologia proposta.

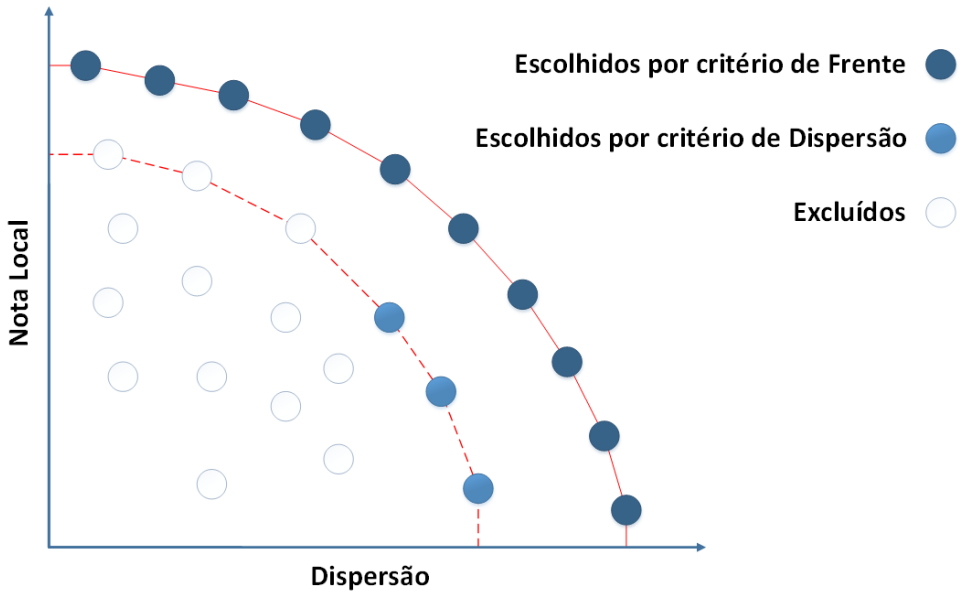


Figura 24: Indivíduos em uma mesma frente não dominada são selecionados por critério de Dispersão.

3.4 Conjunto de Soluções

Com base nas estratégias de competição local e seleção, apresentadas respectivamente nas seções 3.1 e 3.3, o conjunto de soluções final C_f é definido como o conjunto de indivíduos que possuem nota local máxima ao final das execuções. O conjunto C_f representa os indivíduos com maior capacidade funcional dentro de seus próprios nichos, garantindo assim uma boa diversidade entre os resultados. Em um sistema de auxílio à tomada de decisão onde se deve oferecer soluções diversificadas, o conjunto C_f engloba os melhores indivíduos a serem ofertados.

3.5 Metodologia Proposta

A Figura 25 apresenta o diagrama de atividades e modificações implementadas para a estratégia elaborada, combinando PMCNS com a utilização de competição local e dispersão como valores objetivos no NSGA-II.

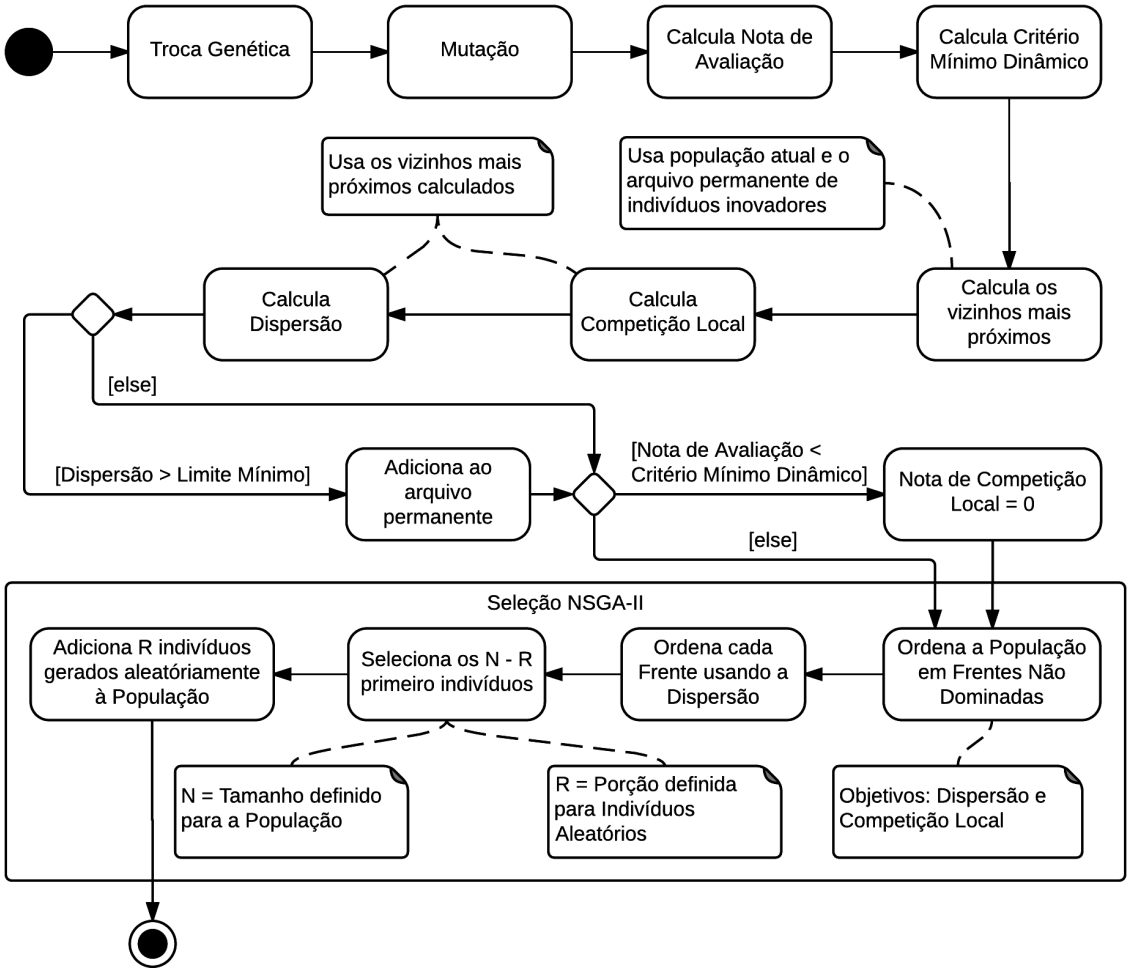


Figura 25: Diagrama de Atividades para o método proposto.

3.6 Evoluindo Mapas para um Jogo de Aventura

Para implementação e testes da técnica proposta, o problema de geração de mapas diferenciados para um jogo de aventura foi o escolhido. Ao gerar novos mapas para um jogo de aventura, buscou-se a criação de mapas que sejam ao mesmo tempo inovadores e interessantes; assim sendo, a estratégia que busca os melhores indivíduos em nichos diferentes se torna promissora para a solução deste tipo de problema.

Um mapa para um jogo de aventura interessante deve apresentar corredores e salas, as quais o jogador deve explorar enquanto combate inimigos e procura a saída para o próximo andar. Para esta pesquisa foram considerados somente o aspecto da exploração do mapa, ignorando quaisquer desafios e obstáculos que possam ser adicionados a um jogo de aventura. Como fator de inovação para os mapas evoluídos, a característica principal analisada foram os aspectos estruturais do mapa, como posicionamento e tamanho das salas e corredores.

3.6.1 Representação do Mapa

Como visto em (CARDAMONE et al., 2011), a representação fenotípica e genotípica dos mapas gerados é um fator importante para a geração automática de conteúdos baseado em busca. A representação deve permitir uma grande variação de conteúdo, ao mesmo tempo que se mantém simples o suficiente para que o algoritmo de busca seja capaz de encontrar indivíduos com boas qualidades.

Os mapas estudados representam uma fase de um jogo de exploração e aventura com visão superior. Sua representação fenotípica é um mapeamento direto da representação do mapa no jogo, definida por uma matriz de dimensões 64 x 64, onde cada entrada representa uma pequena área quadrada do mapa o suficiente para posicionar o personagem do jogador ou qualquer outro elemento, como armadilhas, itens e inimigos. Os possíveis valores para cada entrada na matriz são quatro: vazio, obstáculo, entrada ou saída. As posições vazias do mapa representam corredores e salas, ou seja, áreas por onde o jogador pode navegar livremente, além de possíveis áreas para posicionamento de armadilhas, itens e inimigos. As posições de entrada e saída representam a posição inicial e a posição final para onde o jogador deve caminhar a fim de completar a fase; estas posições devem ser navegáveis entre si. A representação fenotípica foi utilizada para o cálculo de função de avaliação e função de distância. A Figura 26 é a representação de um fenótipo de um mapa com dimensões 16 x 16.

Como representação genotípica utilizada, optou-se pela representação adotada em (LANZI; LOIACONO; STUCCHI, 2014), uma das propostas de representação apresentadas em (CARDAMONE et al., 2011) que gera mapas com espaços reduzidos, característica ideal para um jogo de exploração e aventura. Nesta representação todas as entradas da matriz são, por padrão, obstáculos e somente as entradas vazias são mapeadas de modo a seguir: salas são grandes áreas quadradas do mapa representadas por um terno $< x, y, t >$, onde x e y traduzem a posição central da sala e t o tamanho lateral

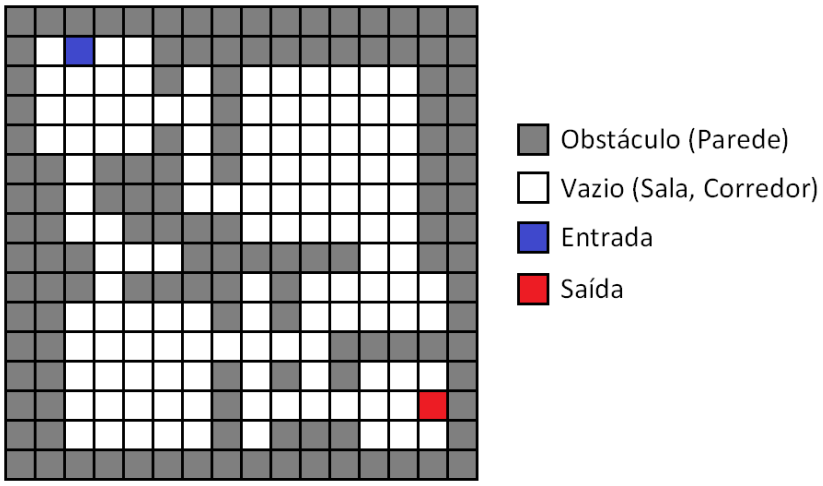


Figura 26: Representação de um fenótipo de mapa um 16 x 16.

da arena; de forma semelhante são descritos os corredores, eles são representados por um terno $< x, y, c >$, onde x e y traduzem a posição central do corredor e c traduz tanto o comprimento quanto a direção do corredor (valores positivos representam corredores horizontais e negativos corredores verticais). Em adição a representação original proposta em (CARDAMONE et al., 2011), definiu-se também a entrada e saída, que são representadas por duas duplas $< x, y >$ que traduzem diretamente suas posições no mapa. A figura 27 apresenta a estrutura da representação genotípica de um mapa qualquer no sistema.

Entrada	Saída	Salas	Corredores
$<x_e,\ y_e>$	$<x_s,\ y_s>$	$<x_1,\ y_1,\ l_1>\ldots\ <x_a,\ y_a,\ l_a>$	$<x_1,\ y_1,\ c_1>\ldots\ <x_b,\ y_b,\ c_b>$

e = Entrada, s = Saída, a = Total de Salas, b = Total de Corredores

Figura 27: Estrutura da representação genotípica de um mapa.

A fim de criar mapas válidos para um jogo de aventura e exploração com o genótipo descrito acima, fez-se necessário a definição de alguns procedimentos adicionais durante a tradução para o fenótipo. Após o posicionamento de todas as salas e corredores, e antes de posicionar a entrada e saída, calcula-se qual é a maior área conexa gerada e removem-se todas as entradas desconexas à esta, fazendo com que voltem a ser consideradas obstáculos. A entrada e saída são posicionadas em seguida, caso estejam em alguma posição considerada como obstáculo são aproximadas através de uma busca radial para a posição vazia mais próxima encontrada. Com estas modificações garante-se que toda a área do mapa é conexa e navegável, certificando a existência de um caminho válido entre a entrada e saída.

Para os experimentos realizados todas as salas possuem tamanho mínimo 10 e máximo 30; os corredores possuem largura 3, comprimento mínimo 3 e máximo 30. Utilizou-se também um número total de 5 salas e 30 corredores, desta forma os mapas gerados possuem algumas grande áreas interligadas por diversos corredores, criando assim um bom ambiente para exploração. Com estas configurações obteve-se, em contraste ao fenótipo de 4096 variáveis, um genótipo final de 109 variáveis (2 para entrada, 2 para saída, 15 para as salas e 90 para os corredores), que possibilita a representação de uma ampla diversidade de mapas ao mesmo tempo permite que o algoritmo de busca encontre indivíduos interessantes.

3.6.2 Função de Avaliação

Naturalmente a escolha da função de avaliação é um fator de extrema importância para o sucesso na execução de algoritmos de busca. Nesta pesquisa, em contraste à outros estudos citados que tentam evoluir mapas interessantes para jogos de tiro (CARDAMONE et al., 2011) (LANZI; LOIACONO; STUCCHI, 2014) ou estratégia (LIAPIS; YANNAKAKIS; TOGELIUS, 2013b) (LIAPIS et al., 2013), a função de avaliação é montada de forma a incentivar a exploração e aventura do jogador, e por este motivo as posições de entrada e saída são de extrema importância. Os mapas são examinados utilizando uma variação de uma das funções de avaliação proposta em (ASHLOCK; LEE; MCGUINNESS, 2011), relativa ao comprimento do menor caminho entre a entrada e saída; quanto mais comprido for o menor caminho existente entre estes, maior o esforço que o jogador precisará realizar a fim de passar para a fase seguinte, contribuindo assim para o fator de exploração e aventura do mapa e fazendo com que ele se torne mais interessante.

Para a função de menor caminho optou-se pelo Algoritmo A*, um algoritmo de busca de menor caminho muito utilizado e conhecido por sua eficiência e precisão. Para a função heurística utilizada escolheu-se a função de Distância de Manhattan, uma vez que os mapas estudados possuem características que se encaixam bem com o uso desta heurística. Permitiu-se também a movimentação nas diagonais no mapa, mas com peso proporcional à diagonal do quadrado; ou seja, o custo de dar um passo na diagonal é maior que o custo de andar na horizontal ou vertical mas ainda é menor que fazer o contorno. Embora permitida a movimentação na diagonal, seguindo o mesmo princípio do paradoxo da conectividade estudado em (RESONFELD; PFALTZ, 1966), não foi permitido que tal movimentação seja possível caso hajam duas paredes nas diagonais adjacentes ao

movimento, pois isso implicaria em uma movimentação não muito realista do jogador atravessando algumas paredes. A Figura 28 mostra as possíveis movimentações do jogador descritas acima.

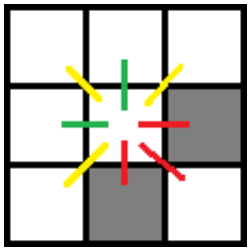


Figura 28: Representação das possíveis movimentações do jogador: em verde movimentações com peso normal, em amarelo movimentações com peso maior e em vermelho movimentações não permitidas.

O menor caminho representa o mínimo que o jogador deverá explorar até encontrar a saída do mapa, portanto ao utilizar esta métrica que incentiva menores caminhos mais longos, garantiu-se que o jogador precisará explorar e caminhar maiores áreas do mapa, aumentando não somente a área total explorada mas também a quantidade de desafios que irá enfrentar. A Figura 29 representa um mapa com as configurações utilizadas durante os testes e o seu menor caminho.

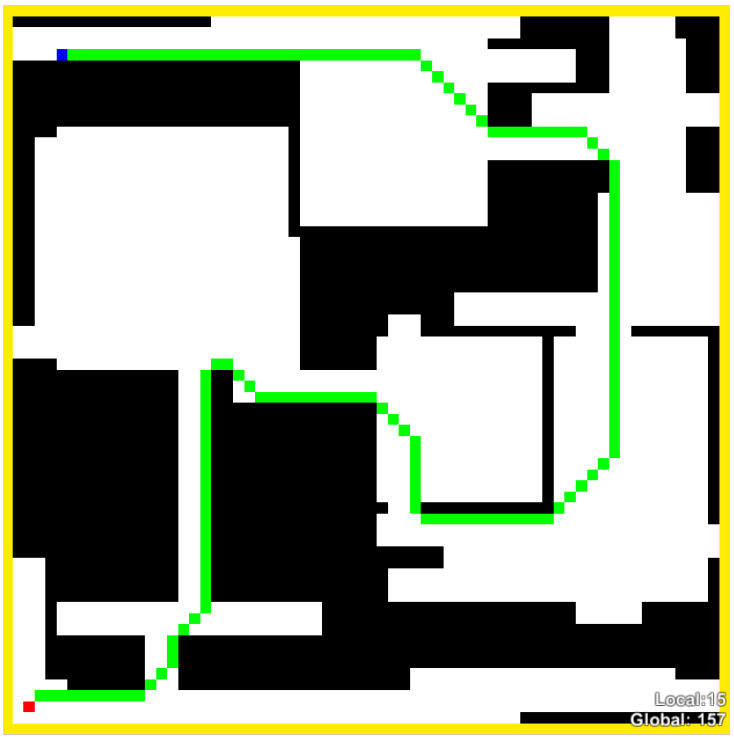


Figura 29: Mapa obtido com Nota de Avaliação 157.

3.6.3 Função Distância

Outra função importante a ser definida é a função de distância, esta função é fortemente relacionada ao contexto do problema e normalmente é utilizada na NS para encontrar indivíduos com comportamentos diferenciados. Nesta pesquisa porém, utilizou-se a característica de exploração e busca de indivíduos diferenciados da NS para encontrar mapas distintos, não havendo um comportamento definido que se relacione com a função de avaliação. Além disso, a escolha da função de distância na estratégia escolhida é de extrema importância para a definição do comportamento dos nichos que serão criados durante a busca, uma vez que esta função é utilizada como principal fator de comparação entre os indivíduos. A má escolha de uma função de distância pode resultar em agrupamentos com características não relevantes ao objetivo, portanto a escolha de uma boa função de distância é crucial para o bom funcionamento da estratégia aplicada.

Para o problema de evolução de mapas em questão, a função de distância definida compara indivíduos estruturalmente diferentes; considerando, por exemplo, posicionamento e tamanho das salas e corredores. Para esta comparação utilizou-se um cálculo de distância puramente fenotípica, uma vez que esta representação está diretamente relacionada às características estruturais finais do mapa.

O cálculo de distância fenotípica é a comparação direta entre o fenótipo dos mapas em questão; ou seja, compara-se diretamente cada uma das 4096 variáveis do fenótipo de dois mapas e tem-se como distância final o número de comparações diferentes entre eles. Desta forma, quando mapas muito parecidos são comparados um baixo valor de distância é obtido, uma vez que muitas de suas variáveis coincidem; de forma oposta, quando mapas com áreas abertas em locais diferentes são comparados o valor de distância calculado é grande.

O valor de diferença entre dois valores x e y quaisquer contidos na matriz de fenótipo de um mapa é dada pela equação 3.2

$$dif(x,y) = \begin{cases} 0 & x = y \\ 1 & x \neq y \end{cases}. \tag{3.2}$$

Sendo assim, a distância final entre dois mapas é calculado pela equação 3.3

$$dist(A,B) = \sum_{i=0}^{m} \sum_{j=0}^{n} dif(A_{ij}, B_{ij}), \tag{3.3}$$

onde A e B são as matrizes de representação do fenótipo de dois mapas distintos e $dif(A_{ij}, B_{ij})$ é o valor de diferença da mesma posição nas duas matrizes calculado de acordo com a equação 3.2.

Devido a natureza da NS, que exige que as distâncias entre todos os elementos da população sejam recalculadas em todas as épocas, este cálculo, embora simples, é um dos grandes gargalos de eficiência do algoritmo, por isso optou-se por realizar todos os cálculos relacionados a distância em *threads* paralelas, otimizando assim a utilização dos núcleos da máquina para o mesmo.

3.7 Diagrama de Classes

A Figura 30 a seguir apresenta o diagrama de classes do pacote computacional desenvolvido para esta pesquisa.

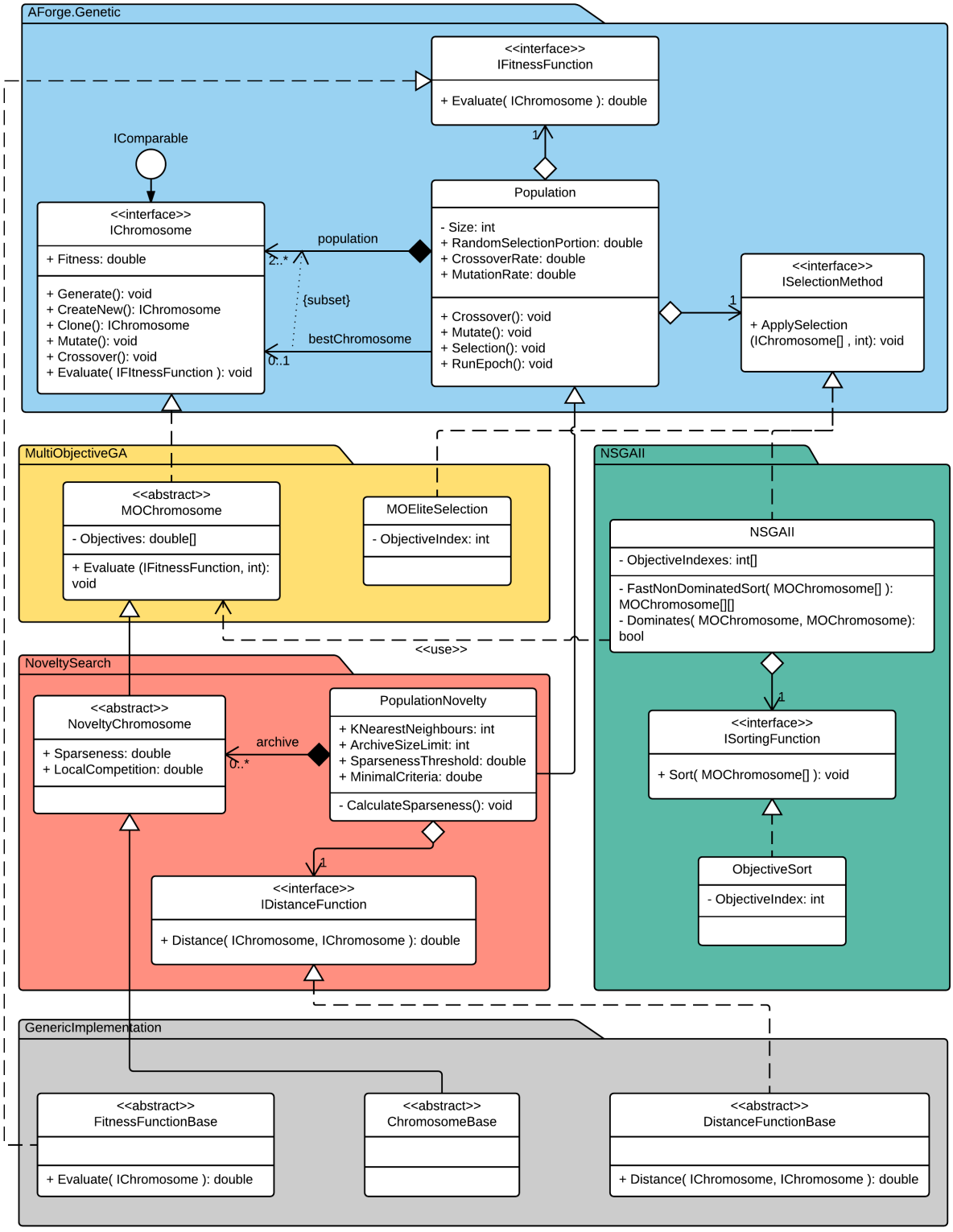


Figura 30: Diagrama de Classes do Pacote Computacional.

3.7.1 Pacote AForge.Genetic

A implementação base do algoritmo genético utilizada nesta pesquisa foi a da AForge.NET (KIRILLOV, 2013), uma *framework* de código aberto desenvolvida em C# para desenvolvedores e pesquisadores nas áreas de Visão Computacional e Inteligência Artificial. A AForge.NET Framework foi escolhida por possuir o código aberto e de fácil modificação; é escrito de forma modular e bastante organizada possibilitando a extensão para as técnicas utilizadas nesta pesquisa.

Pela forma como foi modularizado, o pacote que implementa o algoritmo genético, o AForge.Genetic, foi utilizado tendo sofrido quase nenhuma modificação. As classes e interfaces utilizadas neste pacote são:

- **Population:** Principal responsável pela execução dos métodos do algoritmo genético e manutenção dos indivíduos em cada época.
- **IChromosome:** Interface para a implementação de cromossomos básicos a serem utilizados no algoritmo.
- **IFitnessFunction:** Interface para implementação da função de avaliação.
- **ISelectionMethod:** Interface para implementação do método de seleção que utilizado durante a execução o algoritmo genético.

3.7.2 Pacote MultiObjectiveGA

O pacote MultiObjectiveGA tem como função estender as funções implementadas pela AForge.NET Framework à MOEAs, dando suporte à cromossomos com mais de um valor de objetivo. Nesta pesquisa o algoritmo NSGA-II mantém dois valores de objetivo: a dispersão e o valor de competição local. Mesmo na NS pura é desejável e comum se manter tanto o valor de dispersão quanto o valor da função de avaliação como valores de objetivo.

Este pacote é dependente do pacote AForge.Genetic e possui duas classes:

- **MOChromosome:** Classe abstrata que implementa a interface *IChromosome* do pacote AForge.Genetic e estende sua performance dando suporte à múltiplos valores de objetivo.
- **MOEliteSelection:** Implementação da interface *ISelectionMethod* do pacote AForge.Genetic, que realiza a seleção elitista com base em um dos objetivos previamente especificado.

3.7.3 Pacote NSGAII

O pacote NSGAII possui as classes responsáveis pela implementação do MOEA NSGA-II, dando suporte também à substituição da função de distância de aglomeração, possibilitando a modificação proposta para esta função.

O pacote NSGAII é dependente do Pacote MultiObjectiveGA. São parte deste pacote as seguintes classes e interfaces:

- **NSGAII:** Principal classe de implementação do NSGA-II, que classifica e seleciona os melhores indivíduos com base na frente Pareto ótima, calculada em relação aos objetivos analisados em questão.
- **ISortingFunction:** Interface que expõe a implementação do método de ordenação dentro de uma mesma frente não dominada, originalmente a função de distância de aglomeração.
- **ObjectiveSort:** Implementação da interface *ISortingFunction*, que ordena indivíduos dentro de uma mesma frente não dominada com base em um dos objetivos do *MOChromosome* previamente selecionado. Para a metodologia sugerida utilizou-se o valor de dispersão como objetivo para ordenação.

3.7.4 Pacote NoveltySearch

O pacote NoveltySearch estende e implementa as classes e interfaces necessárias para realização da PMCNS, adicionando suporte ao cálculo de dispersão, manutenção de um arquivo de indivíduos previamente inovadores e expondo variáveis de controle do algoritmo.

Este pacote é dependente dos pacotes AForge.Genetic e MultiObjectiveGA. São parte deste pacote as seguinte classes e interfaces:

- **PopulationNovelty:** Extensão da classe *Population* pertencente ao pacote AForge.Genetic. Implementa os métodos necessários para alterar o algoritmo genético implementado pela AForge.NET Framework e torná-lo uma PMCNS.
- **NoveltyChromosome:** Classe abstrata que estende a classe *MOChromosome* do pacote MultiObjectiveGA. Cromossomo que dá suporte a nota de avaliação, nota de dispersão e nota de competição local. É utilizado pela classe *PopulationNovelty* para execução do PMCNS.

- **IDistanceFunction:** Interface para implementação da função de distância que é utilizada para o cálculo de dispersão durante a NS.

3.7.5 Pacote GenericImplementation

O pacote GenericImplementation contém as classes necessárias para aplicação do sistema no domínio de aplicação escolhido. Para a implementação da solução em qualquer domínio, o único pacote do sistema que deve estendido diretamente é esse, não havendo necessidade de alteração em qualquer outra classe ou interface de outros pacotes. O pacote expõe somente as classes que necessitam serem entendidas e implementadas pelo usuário.

Este pacote é dependente de todos os outros previamente listados e contém três classes abstratas:

- **ChromosomeBase:** Classe abstrata que estende a classe *NoveltyChromosome* do pacote NoveltySearch. Deve ser estendida e implementar o cromossomo do problema em questão.

- **FitnessFunctionBase:** Classe abstrata que implementa a interface *IFitnessFunction* do pacote AForge.Genetic. Deve ser estendida e implementar a Função de Avaliação responsável por mensurar a qualidade dos indivíduos dentro do contexto definido.

- **DistanceFunctionBase:** Classe abstrata que implementa a interface *IDistanceFunction* do pacote NoveltySearch. Deve ser estendida e implementar a função de distância que delimita o contexto dos nichos da busca.

3.7.6 Pacote MapEvolution

A Figura 31 a seguir apresenta o diagrama de classes para o pacote MapEvolution.

O pacote contém as classes necessárias para aplicação do sistema no contexto de geração de mapas diferenciados para um jogo de aventura. Tipicamente para a utilização do sistema em um domínio específico, se faz necessária a criação de um cromossomo, da função de avaliação e da função de distância já expostas pelo pacote GenericImplementation.

São parte deste pacote as seguinte classes:

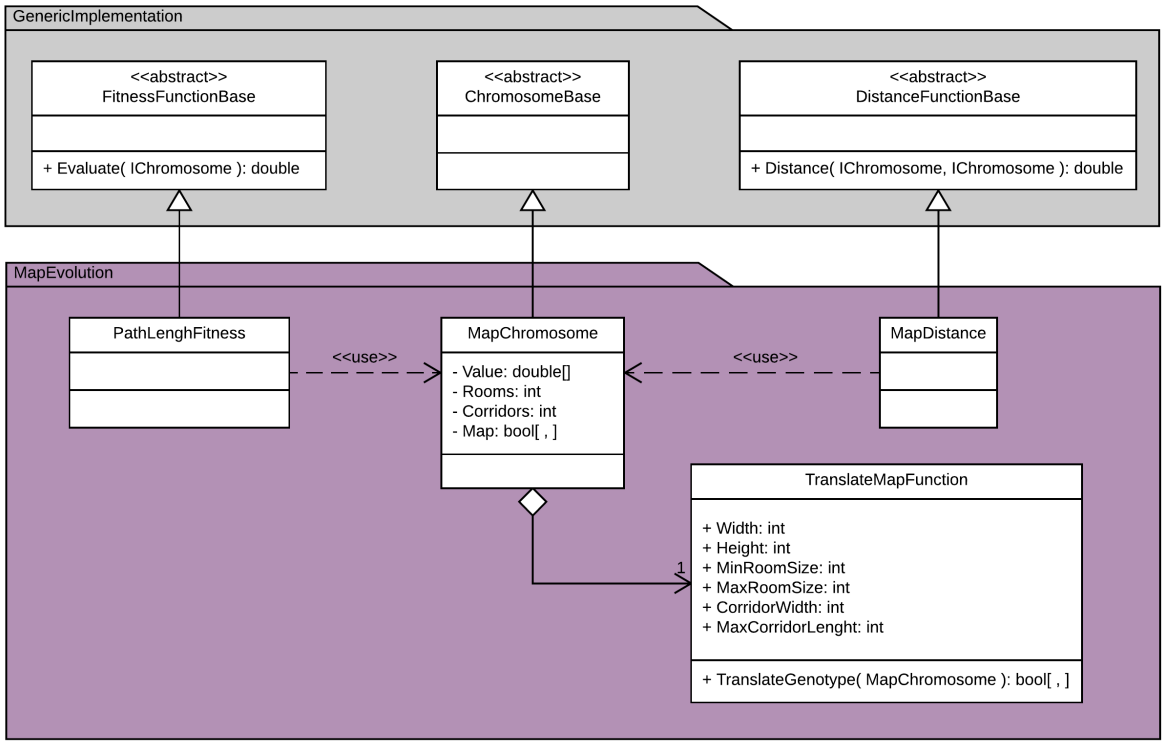


Figura 31: Diagrama de Classes do pacote MapEvolution, extendendo as classes abstratas do pacote GenericImplementation.

- **MapChromosome:** Cromossomo utilizado para descrever mapas de um jogo de aventura. Possui informações relacionadas ao genótipo e fenótipo dos mapas evoluídos, assim como seus métodos de troca genética.

- **TranslateMapFunction:** Classe responsável pela tradução de um genótipo de mapa para seu respectivo fenótipo, baseando-se nas variáveis de controle estabelecidas.

- **PathLenghFitness:** Extensão da classe *FitnessFunctionBase* do pacote GenericImplementation, que calcula a nota de avaliação de um mapa baseando-se em seu fenótipo, utilizando o cálculo de menor caminho para o mesmo.

- **MapDistance:** Extensão da classe *DistanceFunctionBase* do pacote GenericImplementation, que calcula a distância entre dois mapas dentro do espaço de busca escolhido para a avaliação.

4 Experimentos e Resultados

Os experimentos realizados nesta pesquisa evoluem mapas para um jogo de aventura utilizando os métodos e funções previamente elaborados nos capítulos anteriores. O objetivo dos experimentos é testar e comprovar a eficiência do método proposto em obter mapas diversificados e com boas notas de avaliação em comparação à outras técnicas.

Quatro configurações de execuções foram consideradas para os testes: **algoritmo genético (AG)**, **busca inovativa (NS)**, **busca inovativa com competição local (Lehman)** e **busca inovativa com competição local e critério mínimo (Proposta)**.

A primeira configuração utiliza somente a função de avaliação definida para evoluir os mapas da forma tradicional de um algoritmo genético. Esta configuração é utilizada como controle para obtenção de dados base e comparação das notas de avaliação obtidas por outros métodos.

A segunda configuração é a implementação direta da NS, utilizando somente a função de distância definida para a evolução da população. Esta é outra configuração de controle que tem a finalidade de demonstrar que sem uma pressão para adaptação em prol da função de avaliação, os indivíduos gerados, embora bem diversificados, tendem a ter uma nota de avaliação baixa.

A terceira configuração é baseada na implementação proposta em (LEHMAN; STANLEY, 2011b), que utiliza um MOEA (NSGA-II) para combinar a função de dispersão com a função de avaliação local. Desta forma a competição é restringida aos indivíduos próximos em relação ao espaço gerado pela função de dispersão, onde cada indivíduo é recompensado em relação ao número de vezes que sua nota de avaliação é superior à nota de seus vizinhos.

A configuração final é uma modificação proposta para a terceira configuração com a inclusão da ideia de critério mínimo. Nesta configuração, assim como na terceira, utiliza-se um MOEA (NSGA-II) para combinar a função de dispersão com a função de

avaliação local, mas ao invés de utilizar a NS para evoluir a população, ela utiliza uma versão modificada da PMCNS, onde indivíduos que não atingem um critério mínimo, baseado em sua nota de avaliação, recebem uma nota de avaliação local igual a zero. Desta forma indivíduos funcionalmente muito inferiores possuem menores chances durante a seleção, mas ainda podem ser selecionados caso tenham altas notas em relação à função de dispersão, continuando a contribuir com o aumento da diversidade fenotípica.

4.1 Parâmetros de Execução

O tamanho da população em todos os experimentos é de 200 indivíduos, e uma execução corresponde à 200 gerações. A taxa de troca genética foi configurada à 75% e a taxa de mutação à 5% para todas as configurações. As configurações **AG** e **NS** utilizam a seleção elitista com base na função de avaliação e função de dispersão respectivamente, enquanto a seleção nas configurações **Lehman** e **Proposta** são baseadas na seleção utilizada pelo NSGA-II (também elitista) com base na função de dispersão e nota de avaliação local. Em ambos os casos, o NSGA-II utiliza a ordenação por dispersão dentro de cada frente não-ordenada, conforme apresentado previamente.

Enquanto a configuração **AG** é baseada em um algoritmo genético clássico, as configurações **NS**, **Lehman** e **Proposta** utilizam a PMCNS para evoluir suas populações. Nas configurações **NS** e **Lehman** os parâmetros de critério mínimo inicial (cm_0), exigência do critério mínimo (P) e fator de suavização (S) são todos iguais a 0 para obter os mesmos resultados da NS comum; na configuração **Proposta** temos $cm_0 = 0$, $P = 0.5$ e $S = 0.5$ realizando efetivamente a PMCNS com seus valores ideais de acordo com testes realizados em (GOMES; URBANO; CHRISTENSEN, 2012).

Para todas as configurações que utilizam a PMCNS, o limite mínimo de valor de dispersão para que novos indivíduos sejam adicionados ao arquivo é calculado com base na forma descrita em (LEHMAN; STANLEY, 2010) e utilizada também em (GOMES; URBANO; CHRISTENSEN, 2012), onde o parâmetro é inicializado com um valor previamente estipulado por testes e sofre ajustes dinamicamente durante a execução; para os testes realizados nesta pesquisa o valor é inicializado como 1700 e então ajustado dinamicamente: se a cada 5 gerações nenhum novo indivíduo foi adicionado ao arquivo o limite é reduzido em 1%; se mais de 5 indivíduos forem adicionados ao arquivo no mesmo período de gerações, então o valor é incrementado em 5%. Além disso o limite máximo de indivíduos para o arquivo de indivíduos previamente inovadores foi configurado para 500;

caso o número de indivíduos no arquivo seja superior a 500, os indivíduos mais antigos são removidos do arquivo.

O número de k-vizinhos mais próximos utilizados como parâmetro para a PMCNS e cálculo de competição local foi definido como 15 com base em (LEHMAN; STANLEY, 2008) , (LEHMAN; STANLEY, 2011a), (LEHMAN; STANLEY, 2011b), (LEHMAN; STANLEY, 2010), (GOMES; URBANO; CHRISTENSEN, 2012). O valor de competição local, embora não seja utilizado na configuração **NS** para evolução da população, também é calculado durante as execuções e leva em consideração os elementos do arquivo de indivíduos previamente inovadores assim como nas configurações **Lehman** e **Proposta**. Na configuração **AG** entretanto, por não existir uma métrica como o arquivo de indivíduos previamente inovadores, calculou-se os valores de dispersão e competição local baseando-se somente na população resultante ao final das 200 gerações. Nas configurações **AG** e **NS** estes dados foram extraídos somente para fins de comparação. A Tabela 1 apresenta de forma resumida todos os parâmetros utilizados para cada um dos métodos.

Tabela 1: Parâmetros de execução utilizados por cada configuração.

Parâmetro	Algoritmo Genético	Busca Inovativa	Lehman	Proposta
Tam. População	200	200	200	200
Gerações	200	200	200	200
Tx. Troca Genética	75%	75%	75%	75%
Tx. Mutação	5%	5%	5%	5%
Método de Seleção	Elitista (Nota de Avaliação)	Elitista (Dispersão)	NSGA-II (Dispersão + Nota Local)	NSGA-II (Dispersão + Nota Local)
PMCNS - cm_0	N/A	0	0	0
PMCNS - P	N/A	0	0	0.5
PMCNS - S	N/A	0	0	0.5
Disp. Inicial p/ Arquivo	N/A	1700	1700	1700
Freq ajuste	N/A	A cada 5 gerações	A cada 5 gerações	A cada 5 gerações
Min Elements	N/A	0	0	0
Max Elements	N/A	5	5	5
Incremento Limite	N/A	5%	5%	5%
Redução Limite	N/A	1%	1%	1%
k-vizinhos	15*	15	15	15

*Utilizado somente na última geração para fins de controle e comparação

4.2 Métricas

As métricas utilizadas são a Nota Local, Quantidade de Indivíduos em C_f, Nota de Avaliação e Nota de Dispersão; os valores foram obtidos com base na população final P_f após as 200 gerações e em relação ao conjunto C_f de indivíduos que obtiveram a nota local máxima na última geração, como descrito na seção 3.4. Na configuração **AG**, porém, alguns valores para o conjunto C_f não podem ser obtidos; uma vez que, devido a semelhança entre os indivíduos da população final, a nota local máxima possível muitas vezes não é obtida, e o conjunto C_f para tal configuração em muitas execuções acaba sendo vazio. Não sendo possível calcular as médias de valores de conjuntos vazios, as métricas de Nota de Avaliação e Nota de Dispersão não puderam ser calculadas para o conjunto C_f da configuração **AG**.

A primeira métrica é a Nota Local obtida, tanto para o valor médio, quanto para o valor máximo. A Nota Local máxima possível é definida pelo valor k-vizinhos, que restringe e controla diretamente as comparações utilizadas para o cálculo da Nota Local; sendo assim, a Nota Local máxima possível para todas as configurações é 15. A Nota Local média identifica o quão bom são os indivíduos obtidos em relação à seus próprios nichos, quanto maior a nota obtida pela população final, maior a eficiência na exploração de nichos (LEHMAN; STANLEY, 2011b).

A segunda métrica se refere à quantidade de indivíduos contidos no conjunto C_f. Com esta métrica é possível avaliar quantas soluções são geradas por cada uma das configurações em uma única execução. Como o conjunto C_f é composto pelos melhores indivíduos encontrados em cada nicho, quanto mais indivíduos presentes no conjunto, maior a quantidade de soluções diferenciadas a serem oferecidas. Com o objetivo desta pesquisa de oferecer diversas soluções diferenciadas em uma única execução, quanto maior o valor obtido para esta métrica, melhor.

A próxima métrica é em relação a Nota de Avaliação Média, que está diretamente relacionada a função de avaliação definida para o problema; para o contexto de evolução de mapas, esta é o comprimento do caminho mínimo entre a entrada e saída. Os valores são obtidos da média dos indivíduos, tanto para a população final P_f quanto para o conjunto C_f (quando presente). A Nota de Avaliação mínima possível é 0, representando um mapa com entrada e saída sobrepostos, e a máxima é desconhecida e depende das características estruturais definidas para os mapas. Quanto maior o valor obtido, melhor a performance dos indivíduos encontrados pela população.

A última métrica avaliada é referente a Dispersão Média, tanto para a população final P_f quanto para o conjunto C_f de indivíduos. Quando presente, a nota de Dispersão calculada para o conjunto C_f é calculada com base em todos os indivíduos do conjunto, e não somente em relação aos k-vizinhos mais próximos; representando assim a dispersão média total do conjunto de soluções. O valor mínimo de dispersão possível é 0, que ocorre quando dois mapas possuem um fenótipo idêntico; o máximo é 4096, obtida quando dois mapas são completamente distintos, com todas as variáveis de seus fenótipos diferentes. Quanto maior o valor médio obtido, maior a diversidade dos indivíduos finais encontrados.

4.3 Resultados Obtidos para o Jogo Procedural

A seguir são apresentados e comparados os valores obtidos para cada uma das métricas em cada uma das configurações. Para cada uma das 4 configurações, os testes foram executados até que a variação da média acumulada para cada métrica testada em todas as execuções se mantivesse abaixo de 5% por duas execuções seguidas, um número mínimo de 5 execuções foi utilizado. Todas as métricas foram calculadas com base na população final obtida, e o valor obtido para cada métrica é o resultado da média entre todas as execuções realizadas. Para as configurações **AG**, **NS**, **Lehman** e **Proposta**, o número de execuções necessárias para a estabilização da média em todas as métricas foram respectivamente 21, 8, 5 e 7. Os valores obtidos em cada execução são apresentados no Apêndice A. O Apêndice B apresenta exemplos de mapas obtidos no conjunto C_f de cada configuração.

4.3.1 Nota Local

A Tabela 2 e gráfico da Figura 32 a seguir apresentam os valores de Nota Local Média e Máxima obtidas para todas as configurações ao final de suas execuções.

Tabela 2: Valores obtidos por cada configuração para Nota Local.

Métrica	Algoritmo Genético	Busca Inovativa	Lehman	Proposta
Nota Local Média	0.3776	7.9075	9.6340	9.4607
Nota Local Máxima	9.0000	15.0000	15.0000	15.0000

Com base nos dados apresentados é possível notar uma grande variação nos valores obtidos pela configuração **AG** em relação às demais configurações. Esta variação

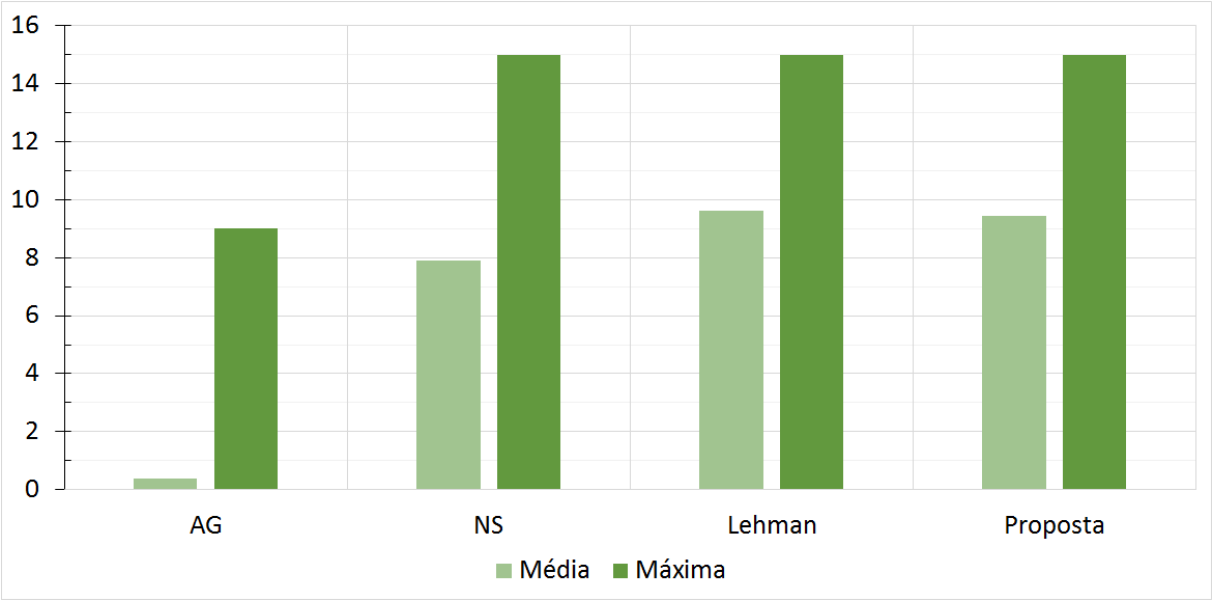


Figura 32: Gráfico dos valores obtidos para Nota Local Média e Máxima.

é o esperado, uma vez que a configuração **AG**, não possui nenhum mecanismo para incentivo à diversidade, gerando assim uma população final com indivíduos muito semelhantes resultando em uma Nota Local baixa; em contrapartida, as demais configurações tem como base a NS, que incentiva a diversidade de seus indivíduos e coopera com a obtenção de maiores valores de Nota Local. Também causa disso, em diversas execuções, a maior nota obtida pela configuração **AG** não foi capaz de atingir a Nota Local máxima possível (15), enquanto nas demais configurações a nota máxima possível foi obtida ao final de todas as execuções.

Dentre as configurações baseadas na NS, a configuração **NS** obteve um valor médio um pouco abaixo em relação às configurações **Lehman** e **Proposta**, apresentando valores aproximadamente 17% abaixo dos obtidos por ambas; a variação entre as duas últimas é inferior a dois por cento. A melhoria em relação a nota média obtida por estas configurações finais, se deve ao fato das modificações propostas em (LEHMAN; STANLEY, 2011b) levarem em consideração a Nota Local durante o processo de seleção, incentivando a preservação de indivíduos com boas Notas Locais e melhorando a média geral da população.

4.3.2 Quantidade de indivíduos em C_f

A Tabela 3 e gráfico da Figura 33 a seguir apresentam a quantidade de indivíduos no conjunto Cf obtidos para todas as configurações ao final de suas execuções.

Tabela 3: Valores obtidos por cada configuração para a Quantidade de indivíduos em C_f.

Métrica	Algoritmo Genético	Busca Inovativa	Lehman	Proposta
Quantidade de indivíduos em C_f	0.4762	12.2500	22.6000	18.4286

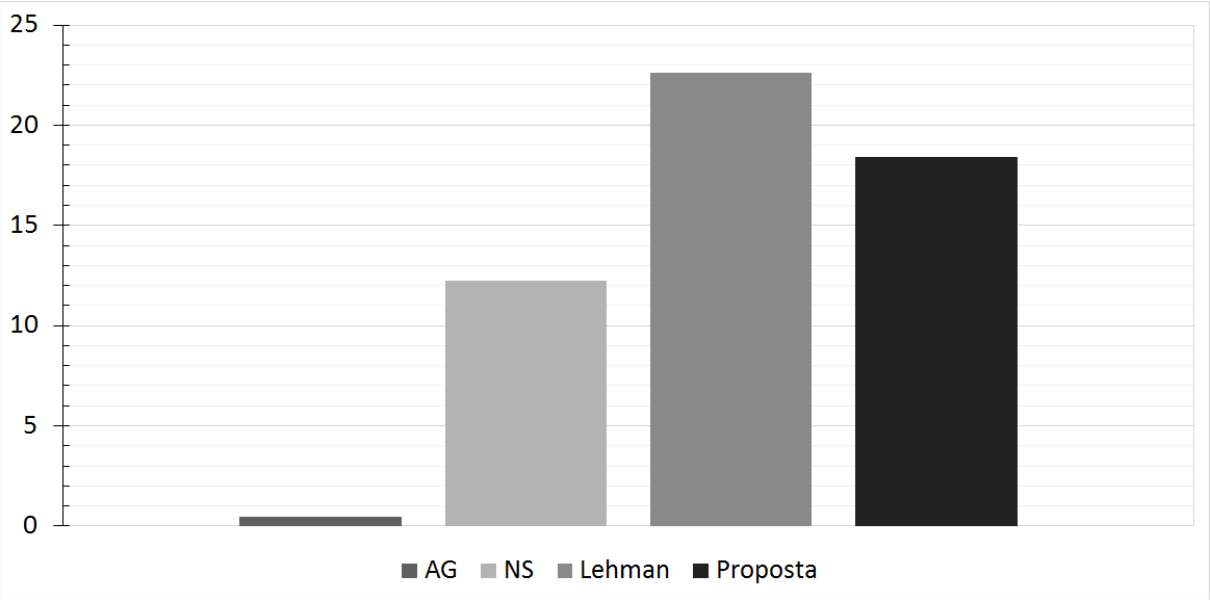


Figura 33: Gráfico da quantidade de indivíduos em C_f.

Assim como para a Nota Loca, e pelos mesmos motivos, há uma grande variação entre os valores obtidos pela configuração **AG** e as demais. Para esta configuração, em diversas execuções a Nota Local máxima possível (15) não foi atingida, resultando em um conjunto C_f vazio; além disso, quando tal nota foi atingida, o mesmo foi realizado por somente alguns indivíduos, contribuindo ainda mais para uma média geral baixa.

Todas as configurações restantes obtiveram valores muito superiores à configuração **AG**; com a configuração **NS**, que obteve a segunda menor quantidade de indivíduos, tendo um valor 2500% superior à mesma. A maior quantidade de indivíduos obtida foi obtida pela configuração **Lehman**, com nota 84,49% superior à configuração **NS**, comprovando a alta capacidade de exploração e separação de nichos, dados pelas modificações propostas em (LEHMAN; STANLEY, 2011b), em comparação com a NS. A configuração **Proposta** obteve resultados 18,46% inferiores em relação a configuração **Lehman**; esta redução se deve ao fato de que a configuração **Proposta** ignora nichos que não possuam alta capacidade funcional, eliminando alguns indivíduos finais pertencentes à nichos de baixa qualidade.

4.3.3 Nota de Avaliação

A Tabela 4 e o gráfico da Figura 34 a seguir apresentam os valores Nota de Avaliação Média obtidos para todas as configurações ao final de suas execuções, tanto para a população final P_f, quanto para os indivíduos do conjunto C_f.

Tabela 4: Valores obtidos por cada configuração para Nota de Avaliação Média em P_f e C_f.

Métrica	Algoritmo Genético	Busca Inovativa	Lehman	Proposta
Nota de Avaliação Média - P_f	243.4019	63.0031	102.2420	133.7679
Nota de Avaliação Média - C_f	N/A	100.7976	134.1249	159.3472

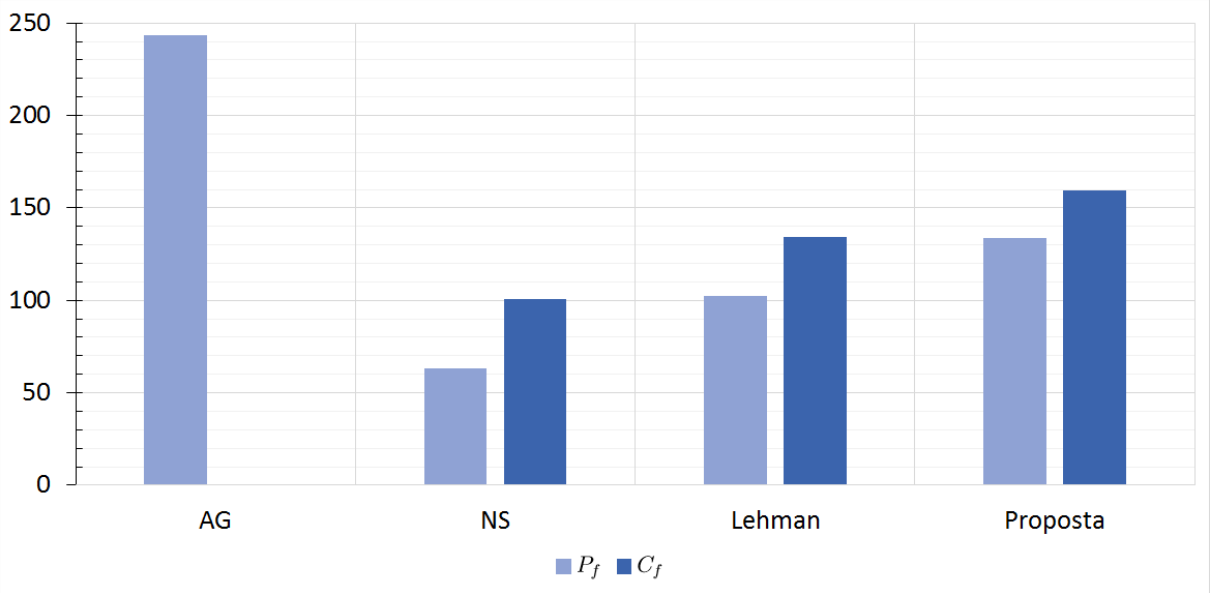


Figura 34: Gráfico das Notas de Avaliação Média obtidas por cada configuração em P_f e C_f.

Os resultados mostram como a configuração **AG** é capaz de obter indivíduos extremamente aptos em relação a Função de Avaliação quando comparado com as outras configurações; entretanto, como discutido na seção 4.2, não é possível definir um conjunto C_f para esta configuração, sendo possível somente a comparação em relação ao valor médio obtido pela população P_f.

A configuração **NS** obteve os piores resultados em relação a Nota de Avaliação, demonstrando a baixa capacidade funcional com o uso da NS sem nenhum mecanismo para recompensar indivíduos funcionais. É importante notar que o uso da NS com outras medidas de comparação, como comportamental por exemplo, é capaz de obter indivíduos mais funcionais (LEHMAN; STANLEY, 2008) (LEHMAN; STANLEY, 2011a); neste estudo entretanto, utiliza-se uma comparação fenotípica que não está conectada a capacidade funcional do indivíduo, incentivando somente a diversificação específica para o qual foi

desenvolvida.

A configuração **Lehman** obteve resultados superiores aos obtidos pela configuração **NS** tanto para P_f quanto para C_f, sendo 62,28% e 33,06% superiores respectivamente. Os resultados obtidos são uma comprovação de que a adição da Função de Avaliação, mesmo que de forma indireta, com a utilização da Competição Local em um MOEA, é capaz de obter indivíduos mais funcionais, embora não seja capaz de atingir a capacidade de um Algoritmo Genético convencional.

A configuração **Proposta**, mesmo tendo atingido somente 54,96% do valor médio obtidos para P_f em relação a configuração **AG**, foi a configuração que obteve melhores resultados dentre as configurações baseadas na NS; em comparação com a configuração **Lehman**, a qual foi utilizada como base para as modificações propostas, as notas médias para P_f e C_f foram respectivamente 30,89% e 18,80% superiores.

4.3.4 Dispersão

A Tabela 5 e o gráfico da Figura 35 a seguir apresentam os valores de Dispersão Média obtidos para todas as configurações ao final de suas execuções, tanto para a população final P_f, quanto para os indivíduos do conjunto C_f.

Tabela 5: Valores obtidos por cada configuração para Dispersão Média em P_f e C_f.

Métrica	Algoritmo Genético	Busca Inovativa	Lehman	Proposta
Dispersão Média - P_f	35.8831	1692.0211	1681.9223	1550.1303
Dispersão Média - C_f	N/A	2044.7265	2034.8144	1971.9663

Novamente, através dos resultados obtidos, é possível notar a grande diferença de valores obtidos pela configuração **AG** em relação às demais configurações. Os valores obtidos para esta configuração comprovam que a população final em Algoritmos Genéticos tende a obter indivíduos extremamente semelhantes, especialmente se comparado com os resultados obtidos pelas configurações que utilizam a NS; essa semelhança justifica tanto a baixa quantidade de indivíduos presentes em C_f quanto os valores obtidos de Nota Local pela configuração. A grande diferença entre os valores demonstra ainda a efetividade da NS em explorar e encontrar indivíduos inovadores.

As configurações **NS** e **Lehman** obtiveram os melhores resultados para a métrica de Dispersão tanto para P_f quanto para C_f, com uma variação inferior à um por cento entre si. A configuração **Proposta**, embora tenha obtido a pior nota para a métrica de dispersão dentre as configurações baseadas na NS, ainda obteve valores somente 7,78%

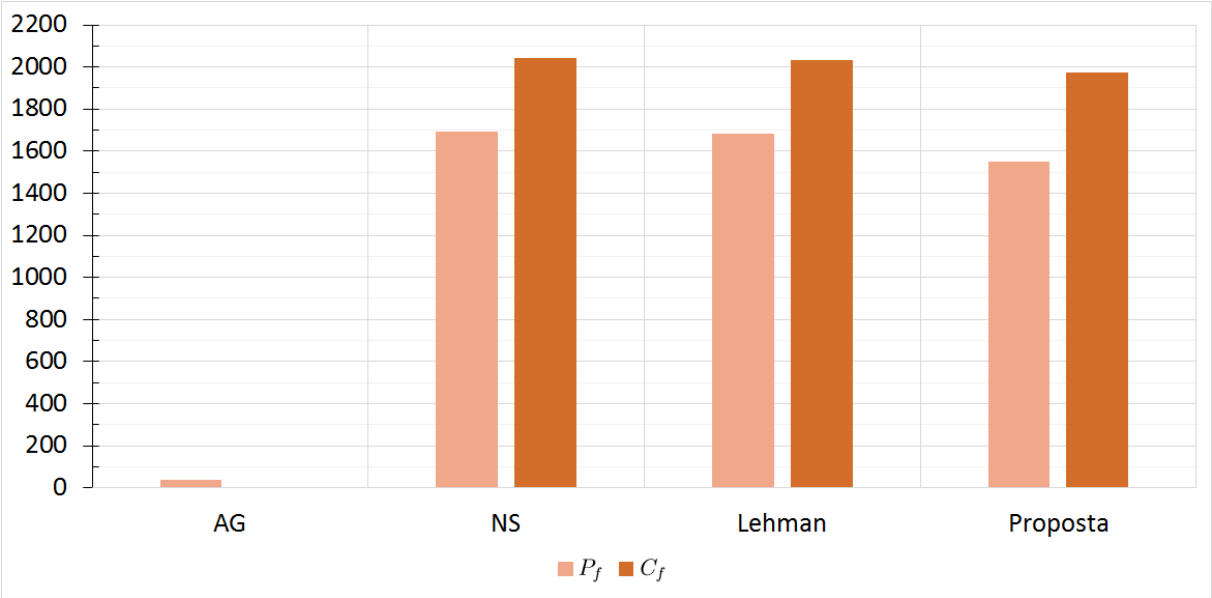


Figura 35: Gráfico da Dispersão Média obtidas por cada configuração em P_f e C_f.

e 3,51% inferiores às notas P_f e C_f da configuração **NS** respectivamente.

4.3.5 Resultados Obtidos para C_f

Para o objetivo desta pesquisa, de obter soluções diversificadas, somente os indivíduos do conjunto C_f são utilizados como solução final. Não sendo capaz de obter indivíduos para o conjunto C_f de forma sólida, como discutido nas seções 4.3.1 e 4.3.2, e avaliando o valor de Dispersão média obtido, apresentado na seção 4.3.4; a configuração **AG** se mostra incapaz de obter soluções que satisfaçam o objetivo proposto. Com isso, uma comparação entre os valores obtidos somente para o conjunto C_f pelas demais configurações se mostra interessante, sendo possível assim analisar dentre as modificações realizadas, quais as vantagens e desvantagens de cada configuração em relação ao objetivo proposto.

O gráfico da Figura 36 apresenta os valores de Dispersão Média e Nota de Avaliação Média obtidos somente pelas configurações baseadas na NS para o conjunto C_f ao final de suas execuções.

Com base no gráfico apresentado na Figura 36, pode-se notar um aumento em relação à métrica de Nota de Avaliação pelas modificações propostas em cada configuração. A implementação baseada nas modificações propostas por **Lehman** obteve um valor 33,06% superior ao valor obtido pela implementação da **NS** pura, enquanto a implementação das modificações propostas nesta pesquisa obteve um valor ainda maior, atingindo

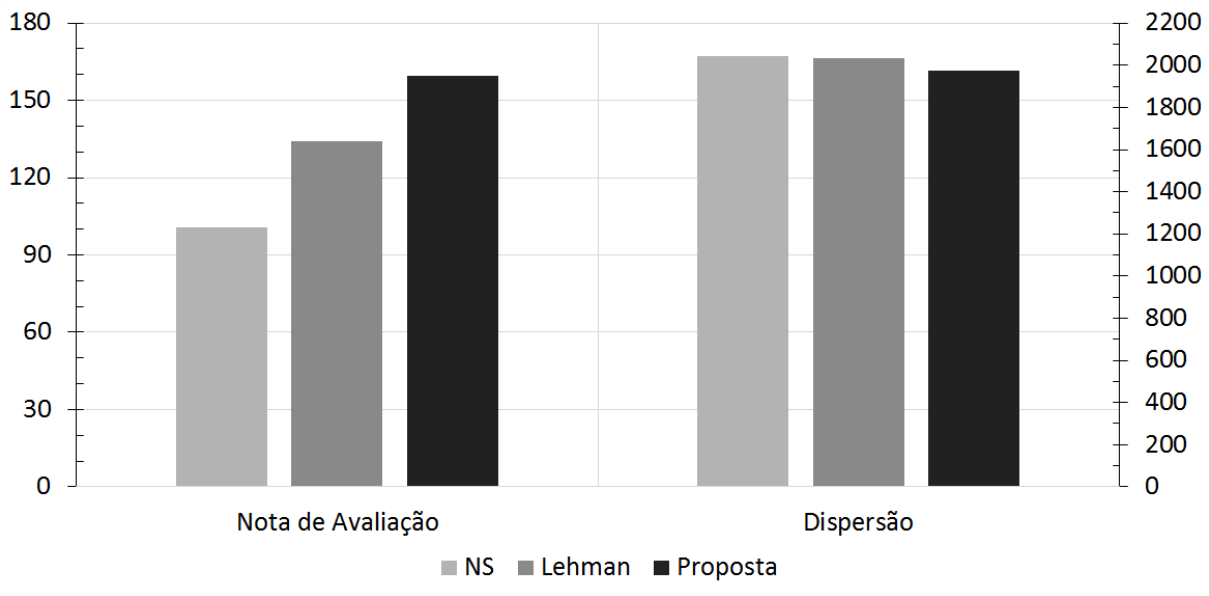


Figura 36: Gráfico de Nota de Avaliação Média e Dispersão Média para os conjuntos C_f obtidos em cada uma das configurações baseadas na NS.

uma nota 18,80% superior a configuração **Lehman** e 58,09% superior a configuração original **NS**.

Analisando os valores obtidos para Dispersão, entretanto, é possível notar uma redução de seus números com a introdução das modificações propostas. Enquanto a configuração **Lehman** obteve somente uma nota de Dispersão 0,48% inferior à configuração **NS**, a configuração **Proposta** obteve uma queda de 3,09% e 3,56% em relação às configurações **NS** e **Lehman** respectivamente. Embora haja uma queda no valor de Dispersão obtido pela configuração proposta, esta queda é mínima se comparada com o ganho obtido pela introdução da NS em relação ao AG, como visto na seção 4.3.4; além disso, o ganho em Nota de Avaliação obtido pela configuração **Proposta** é superior à perda em Dispersão da mesma.

4.4 Tempo de Execução

Os testes foram realizados em uma máquina com processador Intel i7-2700k @3.50GHz (8 CPUs) e memória RAM 8192MB. O tempo de execução para cada configuração foi computado de forma total, considerando todas as execuções necessárias para a obtenção da média descrita na seção 4.3. Desta forma, o tempo de execução médio de cada configuração é obtido pelo tempo total calculado, dividido pela quantidade de execuções realizadas. Para fins de comparação, o consumo de memória durante as execu-

ções também foi observado. A Tabela 6 apresenta o número de execuções; o tempo total obtido, em segundos; o tempo médio calculado; e o consumo de memória médio para cada configuração.

Tabela 6: Tempos e consumo de memória em cada execução obtido pelas configurações.

Métrica	Algoritmo Genético	Busca Inovativa	Lehman	Proposta
Número de Execuções	21	8	5	7
Tempo Total em Segundos	875	1634	1008	1401
Tempo Médio de Execução	41.6667	204.2500	201.6000	200.1429
Consumo Médio de Memória	6.55 MB	19.80 MB	19.30 MB	20.25 MB

Como visto em (CUCCU; GOMEZ, 2011), e verificado pelos valores médios obtidos para tempo de execução e memória, a NS sofre um grande gargalo computacional devido às diversas comparações de distância que devem ser realizadas entre indivíduos da população e do arquivo. Entretanto, se considerarmos a quantidade de indivíduos obtida em média pelas configurações para o conjunto de soluções C_f proposto na seção 3.4, o tempo computacional por solução se torna aceitável. Uma vez que seriam necessárias várias execuções da configuração **AG** para obter o mesmo número de resultados para as demais configurações. A diferença de consumo de memória entre as configurações baseadas na NS não é significativo. A Tabela 7 apresenta o tempo médio de execução por solução obtida em cada configuração.

Tabela 7: Tempo médio de execução por solução obtida em cada configuração.

Métrica	Algoritmo Genético	Busca Inovativa	Lehman	Proposta
Tempo Médio	41.6667	204.2500	201.6000	200.1429
Quantidade de indivíduos em C_f	0.4762	12.2500	22.6000	18.4286
Tempo Médio por Solução	87.4982	16.6735	8.9203	10.8604

Nesta situação, a configuração **AG** é claramente posta em desvantagem devido à estratégia de soluções proposta, mas, mesmo desconsiderando o conjunto C_f, essa tipicamente irá encontrar uma solução por execução. O novo cálculo resultaria em um tempo médio por solução igual a 41,6677 segundos, valor que ainda é inferior aos obtidos por todas as outras configurações.

Os valores revelam ainda, que a configuração **Proposta** possui um bom desempenho em obter uma quantidade diversificada de soluções, sendo inferior somente à configuração **Lehman** devido a quantidade de indivíduos em C_f encontrados pela mesma.

4.5 Discussão das Análises e Testes

Observando-se os resultados obtidos durante os testes é possível verificar a alta capacidade de diversificação introduzida pela NS, com as configurações baseadas na mesma obtendo boas notas nas métrica de Nota Local, Quantidade de indivíduos em C_f e Dispersão. Mesmo a configuração baseada em **AG** tendo obtido os melhores resultados em relação a Nota de Avaliação, esta não se mostrou capaz de gerar resultados bons o suficiente para o objetivo desta pesquisa, resultando em indivíduos muito semelhantes em sua população final.

Os resultados analisados na seção 4.3.5 suportam a hipótese de que, no contexto de exploração de indivíduos inovadores, a introdução de um critério mínimo global para restrição da busca, em adição à estratégia de Competição Local apresentada por (LEHMAN; STANLEY, 2011b), contribui para a melhoria da qualidade funcional dos indivíduos obtidos. Mesmo havendo uma relação de troca em relação à Nota de Avaliação e Dispersão, esta representa um ganho de 18,80 e 58,09 % para Nota de Avaliação contra uma queda de apenas 3,09 e 3,56 % para Dispersão em relação as configurações **NS** e **Lehman** respectivamente.

5 Conclusão

Este trabalho apresentou e implementou um algoritmo evolutivo capaz de obter indivíduos diversificados e com alta capacidade funcional em uma única execução. O mesmo, com base em uma função de distância fenotípica, é capaz de manter diferentes nichos morfológicos durante a busca ao mesmo tempo que seleciona os melhores indivíduos dentro de cada um deles. Além disso, o algoritmo é eficaz em eliminar nichos com baixa capacidade funcional da busca, obtendo um conjunto final de indivíduos com melhor nota de avaliação quando comparado às técnicas nas quais se baseia.

As implementações e testes foram conduzidos dentro do contexto de geração procedural de mapas para jogos de aventura. Para tal, foram definidos o mapeamento entre genótipo e fenótipo, assim como as funções de avaliação e distância. Os testes foram realizados utilizando quatro diferentes configurações: AG, NS, Lehman e Proposta; conforme descritas no capítulo 4.

Os resultados obtidos permitiram confirmar os benefícios da introdução da NS no contexto de busca por inovação, com todas as configurações baseadas na NS obtendo valores de dispersão muito superiores aos valores obtidos pela configuração baseada em AG, demonstrando que a introdução da mesma contribui significativamente para a obtenção de indivíduos diferenciados. Embora haja um grande ganho em diversificação, os testes revelam que a utilização da NS sem uma função de distância que se relacione de algum modo com a performance, não contribui com a busca por boas notas de avaliação, selecionando indivíduos somente com base na morfologia para a qual foi elaborada.

Os resultados mostram ainda, que as modificações realizadas no algoritmo apresentado em (LEHMAN; STANLEY, 2011b), que foram aqui introduzidas e dão suporte a ideia de exclusão de nichos de baixa capacidade, obtiveram bons resultados em relação à qualidade dos indivíduos encontrados. Quando comparados na seção 4.5, os resultados obtidos pela configuração Lehman e Proposta, revelam um ganho de 18,80% em relação à nota de avaliação média do conjunto C_f obtido pela segunda, ao mesmo tempo

que mantém o valor de dispersão com uma variação de apenas -3,09%. Estes valores comprovam a eficiência da remoção de nichos ruins da busca para a melhoria da qualidade funcional obtida, dando prioridade a nichos com alta capacidade durante a seleção, sem que a diversidade introduzida pela NS seja prejudicada.

Nos testes realizados, a configuração Proposta foi capaz de obter, de forma fidedigna, uma quantidade significativa de mapas como solução em todas as execuções. Além disso, se comparados com os mapas obtidos pela AG, a configuração Proposta demonstra ser capaz de, em uma única execução, encontrar mapas visualmente e estruturalmente bem diversificados. Outra consequência disso, conforme discutido na seção 4.4, é que mesmo obtendo um tempo total de execução maior que a AG, o tempo médio por solução obtida pela proposta também se torna um avanço em relação a capacidade computacional da solução do problema proposto.

Com isso, as técnicas e algoritmos aqui propostos, evidenciam ser viáveis para o contexto de geração procedural de conteúdo onde se deseja evitar a baixa qualidade dos itens gerado, assim como repetição ou inconsistência nos padrões dos mesmos. De forma análoga, as técnicas aqui apresentadas também podem ser de boa utilidade para geração de opções diferentes em um sistema de auxílio à decisão.

Este trabalho gerou como produto uma ferramenta funcional para geração de mapas para jogos de aventura, que pode ser utilizado tanto para criação de conteúdo dinâmico quanto para auxílio no processo de *Level Design*.

Fazem parte da contribuição gerada por este trabalho: o pacote computacional genérico da ferramenta proposta; assim como as extensões para utilização da NS, MOEAs e NSGA-II para a *framework* AForge.NET (KIRILLOV, 2013), com código fonte disponível online em <https://github.com/lexmelotti/nichesearch> sob licença GNU LGPLv3.

5.1 Trabalhos Futuros

Como sugestão de estudos de continuidade, propõe-se uma extensão do algoritmo apresentado que aprimore a busca dentro dos nichos obtidos, efetuando uma pesquisa por indivíduos com altas notas de avaliação limitada por barreiras, evitando assim a convergência global da busca e mantendo a diversidade obtida pelo algoritmo aqui apresentado. Para tal, sugere-se a utilização de um algoritmo genético com barreiras. Utilizando os indivíduos encontrados em C_f como base para o particionamento, e técnicas de agrupamento como *k-means* e *k-medoids*, a busca pode ser delimitada por regiões definidas

pelas medianas dos centróides e medóides obtidos. Outra opção possível, uma vez que o algoritmo aqui apresentado já realiza uma segregação por nichos morfológicos, é limitar a busca ao redor dos indivíduos de C_f com o cálculos de medianas entre os mesmos.

Outra sugestão de continuidade é a utilização de operadores dinâmicos genéticos e de dispersão, com alteração baseada na análise dos resultados obtidos ao longo das gerações.

Sugere-se ainda a utilização da estrutura de dados *QuadTrees* em trabalhos futuros para otimização do cálculo da busca por vizinhos mais próximos, assim como a possibilidade da utilização de outras métricas para o cálculo de distância entre mapas, podendo considerar a utilização de técnicas de similaridade de imagens, comparação entre os caminhos gerados, e outros cálculos utilizando o genótipo para simplificação e otimização.

Referências

ACTIVISION. *Pitfall!* 1982. Activision.

ACTIVISION. *River Raid*. 1982. Activision.

ANGELIDES, M. C.; AGIUS, H. Procedural content generation. In: *Handbook of Digital Games*. [S.l.]: Wiley-IEEE Press, 2014. p. 62–91.

ASHLOCK, D.; LEE, C.; MCGUINNESS, C. Search-based procedural generation of maze-like levels. *IEEE Transactions on Computational Intelligence and AI in Games*, IEEE, v. 3, n. 3, p. 260–273, 2011.

AYCOCK, J. Procedural content generation. In: *Retrogame Archeology*. [S.l.]: Springer, 2016. p. 109–143.

BETHESTA GAME STUDIOS. *The Elder Scrolls IV: Oblivion*. 2006. Bethesda Softworks.

BLIZZARD ENTERTAINMENT. *StarCraft*. 1998. Blizzard Entertainment.

BLIZZARD NORTH. *Diablo*. 1996. Blizzard Entertainment.

BRABEN, D.; BELL, I. *Elite*. 1984. Acornsoft.

CARDAMONE, L.; LOIACONO, D.; LANZI, P. L. Interactive evolution for the procedural generation of tracks in a high-end racing game. In: ACM. *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. [S.l.], 2011. p. 395–402.

CARDAMONE, L. et al. Evolving interesting maps for a first person shooter. In: SPRINGER. *European Conference on the Applications of Evolutionary Computation*. [S.l.], 2011. p. 63–72.

CUCCU, G.; GOMEZ, F. When novelty is not enough. In: SPRINGER. *European Conference on the Applications of Evolutionary Computation*. [S.l.], 2011. p. 234–243.

DEB, K. et al. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, IEEE, v. 6, n. 2, p. 182–197, 2002.

EVOLUTIONARY GAMES. *Galactic Arms Race*. 2012. Evolutionary Games.

GEARBIX SOFTWARE. *Borderlands*. 2009. 2K Games.

GOMES, J.; URBANO, P.; CHRISTENSEN, A. L. Progressive minimal criteria novelty search. In: SPRINGER. *Ibero-American Conference on Artificial Intelligence*. [S.l.], 2012. p. 281–290.

HASTINGS, E. J.; GUHA, R. K.; STANLEY, K. O. Automatic content generation in the galactic arms race video game. *IEEE Transactions on Computational Intelligence and AI in Games*, IEEE, v. 1, n. 4, p. 245–263, 2009.

HASTINGS, E. J.; GUHA, R. K.; STANLEY, K. O. Evolving content in the galactic arms race video game. In: IEEE. *2009 IEEE Symposium on Computational Intelligence and Games*. [S.l.], 2009. p. 241–248.

HENDRIKX, M. et al. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, ACM, v. 9, n. 1, p. 1, 2013.

KERSSEMAKERS, M. et al. A procedural procedural level generator generator. In: IEEE. *2012 IEEE Conference on Computational Intelligence and Games (CIG)*. [S.l.], 2012. p. 335–341.

KIRILLOV, A. *Aforge. net framework*. 2013.

LANZI, P. L.; LOIACONO, D.; STUCCHI, R. Evolving maps for match balancing in first person shooters. In: IEEE. *2014 IEEE Conference on Computational Intelligence and Games*. [S.l.], 2014. p. 1–8.

LEHMAN, J.; STANLEY, K. O. Exploiting open-endedness to solve problems through the search for novelty. In: *ALIFE*. [S.l.: s.n.], 2008. p. 329–336.

LEHMAN, J.; STANLEY, K. O. Revising the evolutionary computation abstraction: minimal criteria novelty search. In: ACM. *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. [S.l.], 2010. p. 103–110.

LEHMAN, J.; STANLEY, K. O. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, MIT Press, v. 19, n. 2, p. 189–223, 2011.

LEHMAN, J.; STANLEY, K. O. Evolving a diversity of virtual creatures through novelty search and local competition. In: ACM. *Proceedings of the 13th annual conference on Genetic and evolutionary computation*. [S.l.], 2011. p. 211–218.

LIAPIS, A. et al. Adaptive game level creation through rank-based interactive evolution. In: IEEE. *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*. [S.l.], 2013. p. 1–8.

LIAPIS, A.; YANNAKAKIS, G. N.; TOGELIUS, J. Enhancements to constrained novelty search: Two-population novelty search for generating game content. In: ACM. *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. [S.l.], 2013. p. 343–350.

LIAPIS, A.; YANNAKAKIS, G. N.; TOGELIUS, J. Generating map sketches for strategy games. In: SPRINGER. *European Conference on the Applications of Evolutionary Computation*. [S.l.], 2013. p. 264–273.

LIAPIS, A.; YANNAKAKIS, G. N.; TOGELIUS, J. Sentient world: Human-based procedural cartography. In: SPRINGER. *International Conference on Evolutionary and Biologically Inspired Music and Art*. [S.l.], 2013. p. 180–191.

LIAPIS, A.; YANNAKAKIS, G. N.; TOGELIUS, J. Constrained novelty search: A study on game content generation. *Evolutionary computation*, MIT Press, v. 23, n. 1, p. 101–129, 2015.

LINDEN, R. van der; LOPES, R.; BIDARRA, R. Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*, IEEE, v. 6, n. 1, p. 78–89, 2014.

MAXIS. *Spore*. 2008. Electronic Arts.

MIYAMOTO, S. *Super Mario Bros.* 1985. Nintendo.

MOJANG. *Minecraft*. 2011. Mojang.

RE-LOGIC. *Terraria*. 2011. Re-Logic.

RESONFELD, A.; PFALTZ, J. L. *Sequential operations in Digital Image Processing*. [S.l.]: JACM, 1966.

RISI, S. et al. Combining search-based procedural content generation and social gaming in the petalz video game. In: CITESEER. *Aiide*. [S.l.], 2012.

TOGELIUS, J. et al. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, IEEE, v. 3, n. 3, p. 172–186, 2011.

TOY, M. et al. *Rogue*. 1980. Auto-publicado.

VALTCHANOV, V.; BROWN, J. A. Evolving dungeon crawler levels with relative placement. In: ACM. *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering*. [S.l.], 2012. p. 27–35.

VALVE SOUTH. *Left 4 Dead*. 2008. Valve Corporation.

WOOLLEY, B. G.; STANLEY, K. O. A novel human-computer collaboration: combining novelty search with interactive evolution. In: ACM. *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. [S.l.], 2014. p. 233–240.

YANNAKAKIS, G. N.; TOGELIUS, J. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing*, IEEE, v. 2, n. 3, p. 147–161, 2011.

YU, D. *Spelunky*. 2012. Mossmouth.

APÊNDICE A – Tabelas dos Valores Obtidos nos Testes

As tabelas a seguir apresentam o Valor Obtido (v_i), o Valor Médio Acumulado (VMA_i) e a Variação de Valor Médio Acumulado (Var_i) em todas as métricas, para cada uma das execuções realizadas. As iterações encerram quando todas as métricas obtém Variação de Valor Médio Acumulado (Var_i) com valor igual ou inferior a 5% (0.05) em duas iterações seguidas. O número mínimo de 5 execuções foi utilizado.

A Tabela abaixo é um modelo de representação das tabelas a seguir.

Execução	Nota de Avaliação Média	Dispersão Média	Nota Local Média	Nota Local Máxima	C_f Count	Nota de Avaliação em C_f	Dispersão em C_f
1	Valores Obtido nesta Execução (v) Valores Médios Acumulados (VMA) Variação de Valor Médio Acumulado em relação à execução anterior (Var)						
...							
i	v_i $VMA_i = (v_1 + v_2 + ... + v_{i-1} + v_i)/i$ $Var_i = (VMA_i/VMA_{i-1}) - 1$						
...							
n-1	v_{n-1} VMA_{n-1} $\lvert Var_{n-1} \rvert \leq 0.05$						
n	v_n VMA_n $\lvert Var_n \rvert \leq 0.05$						

Variação abaixo de 5%

Valores utilizados

A.1 Algoritmo Genético

Os valores de Nota de Avaliação em C_f e Dispersão em C_f não se aplicam à configuração **AG** conforme discutido na seção 4.2.

Execução	Nota de Avaliação Média	Dispersão Média	Nota Local Média	Nota Local Máxima	C_f Count	Nota de Avaliação em C_f	Dispersão em C_f
1	255.3900	43.0220	2.1450	15.0000	1.0000	-	-
	255.3900	43.0220	2.1450	15.0000	1.0000	-	-
	-	-	-	-	-	-	-
2	292.3300	21.9953	0.5250	6.0000	0.0000	-	-
	258.8600	32.5087	1.3350	10.5000	0.5000	-	-
	0.1485	-0.2444	-0.3776	-0.3000	-0.5000	-	-
3	263.7850	17.9270	0.8650	10.0000	0.0000	-	-
	260.5017	27.6481	1.1783	10.3333	0.3333	-	-
	0.0063	-0.1495	-0.1174	-0.0159	-0.3333	-	-
4	237.0050	28.4870	0.0750	15.0000	1.0000	-	-
	254.6275	27.8578	0.9025	11.5000	0.5000	-	-
	-0.0225	0.0076	-0.2341	0.1129	0.5000	-	-
5	260.0000	10.6010	0.0000	0.0000	0.0000	-	-
	255.7020	24.4065	0.7220	9.2000	0.4000	-	-
	0.0042	-0.1239	-0.2000	-0.2000	-0.2000	-	-
6	241.0000	118.8943	0.0000	0.0000	0.0000	-	-
	253.2517	40.1544	0.6017	7.6667	0.3333	-	-
	-0.0096	0.6452	-0.1667	-0.1667	-0.1667	-	-
7	208.2350	40.6240	0.1500	15.0000	2.0000	-	-
	246.8207	40.2215	0.5371	8.7143	0.5714	-	-
	-0.0254	0.0017	-0.1072	0.1366	0.7143	-	-
8	226.1050	31.7023	0.2800	15.0000	1.0000	-	-
	244.2313	39.1566	0.5050	9.5000	0.6250	-	-
	-0.0105	-0.0265	-0.0598	0.0902	0.0938	-	-
9	277.3150	38.7190	0.7550	14.0000	0.0000	-	-
	247.9072	39.1080	0.5328	10.0000	0.5556	-	-
	0.0151	-0.0012	0.0550	0.0526	-0.1111	-	-
10	228.0000	29.9807	0.0000	0.0000	0.0000	-	-
	245.9165	38.1953	0.4795	9.0000	0.5000	-	-
	-0.0080	-0.0233	-0.1000	-0.1000	-0.1000	-	-
11	240.0050	29.3243	0.0750	15.0000	1.0000	-	-
	245.3791	37.3888	0.4427	9.5455	0.5455	-	-
	-0.0022	-0.0211	-0.0767	0.0606	0.0909	-	-
12	271.7650	62.5497	1.6800	15.0000	3.0000	-	-
	247.5779	39.4865	0.5458	10.0000	0.7500	-	-
	0.0090	0.0561	0.2329	0.0476	0.3750	-	-
13	235.0000	17.3193	0.0000	0.0000	0.0000	-	-
	246.6104	37.7805	0.5038	9.2308	0.6923	-	-
	-0.0039	-0.0432	-0.0769	-0.0769	-0.0769	-	-
14	233.0050	10.1877	0.0750	15.0000	1.0000	-	-
	245.6386	35.8095	0.4732	9.6429	0.7143	-	-
	-0.0039	-0.0522	-0.0608	0.0446	0.0317	-	-

Execução	Nota de Avaliação Média	Dispersão Média	Nota Local Média	Nota Local Máxima	C_f Count	Nota de Avaliação em C_f	Dispersão em C_f
15	217.0000	48.4733	0.0000	0.0000	0.0000	-	-
	243.7293	36.6538	0.4417	9.0000	0.6667	-	-
	-0.0078	0.0236	-0.0667	-0.0667	-0.0667	-	-
16	232.0750	49.5183	0.3600	13.0000	0.0000	-	-
	243.0009	37.4578	0.4366	9.2500	0.6250	-	-
	-0.0030	0.0219	-0.0116	0.0278	-0.0625	-	-
17	256.0000	43.4033	0.0000	0.0000	0.0000	-	-
	243.7656	37.8076	0.4109	8.7059	0.5882	-	-
	0.0031	0.0093	-0.0588	-0.0588	-0.0588	-	-
18	227.0000	44.6817	0.0000	0.0000	0.0000	-	-
	242.8342	38.1895	0.3881	8.2222	0.5556	-	-
	-0.0038	0.0101	-0.0556	-0.0556	-0.0556	-	-
19	243.0300	10.6437	0.1950	13.0000	0.0000	-	-
	242.8445	36.7397	0.3779	8.4737	0.5263	-	-
	0.0000	-0.0380	-0.0262	0.0306	-0.0526	-	-
20	250.3850	37.5533	0.6100	14.0000	0.0000	-	-
	243.2215	36.7804	0.3895	8.7500	0.5000	-	-
	0.0016	0.0011	0.0307	0.0326	-0.0500	-	-
21	247.0100	17.9377	0.1400	14.0000	0.0000	-	-
	243.4019	35.8831	0.3776	9.0000	0.4762	-	-
	0.0007	-0.0244	-0.0305	0.0286	-0.0476	-	-

Tempo total de execução: 875 segundos

A.2 Busca Inovativa

Execução	Nota de Avaliação Média	Dispersão Média	Nota Local Média	Nota Local Máxima	C_f Count	Nota de Avaliação em C_f	Dispersão em C_f
1	55.7400	1695.3943	7.9950	15.0000	14.0000	90.6429	2028.8901
	55.7400	1695.3943	7.9950	15.0000	14.0000	90.6429	2028.8901
	-	-	-	-	-	-	-
2	74.2200	1692.8953	8.1450	15.0000	14.0000	105.1429	2032.7143
	64.9800	1694.1448	8.0700	15.0000	14.0000	97.8929	2030.8022
	0.1658	-0.0007	0.0094	0.0000	0.0000	0.0800	0.0009
3	62.9100	1702.1457	7.7950	15.0000	11.0000	109.1818	2053.8182
	64.2900	1696.8118	7.9783	15.0000	13.0000	101.6558	2038.4742
	-0.0106	0.0016	-0.0114	0.0000	-0.0714	0.0384	0.0038
4	55.1800	1679.0587	7.7450	15.0000	8.0000	106.1250	2051.1429
	62.0125	1692.3735	7.9200	15.0000	11.7500	102.7731	2041.6414
	-0.0354	-0.0026	-0.0073	0.0000	-0.0962	0.0110	0.0016
5	65.0550	1697.0447	7.5400	15.0000	10.0000	103.1000	2049.3778
	62.6210	1693.3077	7.8440	15.0000	11.4000	102.8385	2043.1886
	0.0098	0.0006	-0.0096	0.0000	-0.0298	0.0006	0.0008
6	70.8900	1691.3567	8.4750	15.0000	15.0000	104.0000	2041.7905
	63.9992	1692.9826	7.9492	15.0000	12.0000	103.0321	2042.9556
	0.0220	-0.0002	0.0134	0.0000	0.0526	0.0019	-0.0001
7	52.7400	1685.8547	7.9600	15.0000	11.0000	88.4545	2050.5636
	62.3907	1691.9643	7.9507	15.0000	11.8571	100.9496	2044.0425
	-0.0251	-0.0006	0.0002	0.0000	-0.0119	-0.0202	0.0005
8	67.2900	1692.4190	7.6050	15.0000	15.0000	99.7333	2049.5143
	63.0031	1692.0211	7.9075	15.0000	12.2500	100.7976	2044.7265
	0.0098	0.0000	-0.0054	0.0000	0.0331	-0.0015	0.0003

Tempo total de execução: 1634 segundos

A.3 Competição Local

Execução	Nota de Avaliação Média	Dispersão Média	Nota Local Média	Nota Local Máxima	C_f Count	Nota de Avaliação em C_f	Dispersão em C_f
	102.0000	1675.3643	9.9100	15.0000	25.0000	129.6400	2037.5533
1	102.0000	1675.3643	9.9100	15.0000	25.0000	129.6400	2037.5533
	-	-	-	-	-	-	-
	100.2050	1679.6123	9.3900	15.0000	22.0000	139.5455	2028.9134
2	101.1025	1677.4883	9.6500	15.0000	23.5000	134.5927	2033.2334
	-0.0088	0.0013	-0.0262	0.0000	-0.0600	0.0382	-0.0021
	103.5150	1683.0843	9.7500	15.0000	16.0000	129.1250	2050.2750
3	101.9067	1679.3537	9.6833	15.0000	21.0000	132.7702	2038.9139
	0.0080	0.0011	0.0035	0.0000	-0.1064	-0.0135	0.0028
	101.5450	1684.4330	9.5450	15.0000	24.0000	137.5833	2027.3949
4	101.8163	1680.6235	9.6488	15.0000	21.7500	133.9734	2036.0342
	-0.0009	0.0008	-0.0036	0.0000	0.0357	0.0091	-0.0014
	103.9450	1687.1173	9.5750	15.0000	26.0000	134.7308	2029.9354
5	102.2420	1681.9223	9.6340	15.0000	22.6000	134.1249	2034.8144
	0.0042	0.0008	-0.0150	0.0000	0.0391	0.0011	-0.0006

Tempo total de execução: 1008 segundos

A.4 Proposta

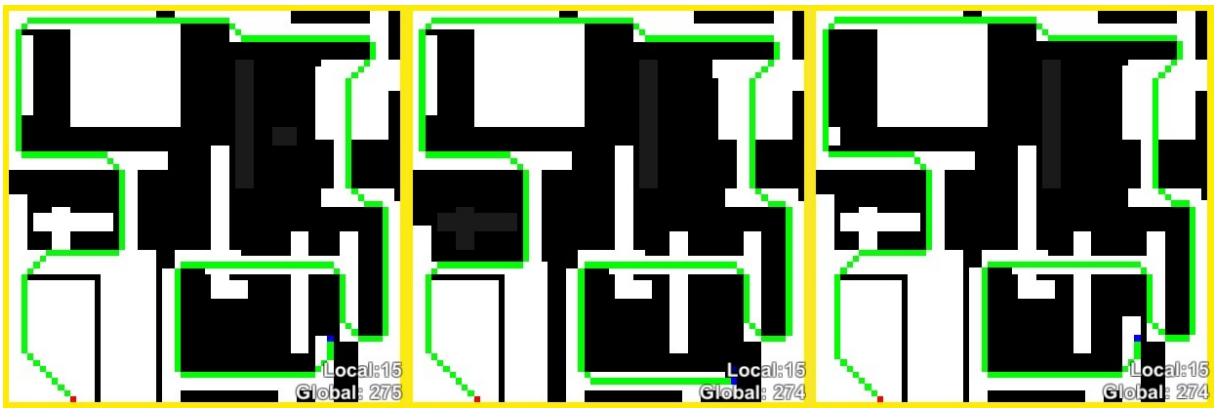
Execução	Nota de Avaliação Média	Dispersão Média	Nota Local Média	Nota Local Máxima	C_f Count	Nota de Avaliação em C_f	Dispersão em C_f
1	131.5750	1595.9700	9.4700	15.0000	17.0000	158.5294	1971.8824
	131.5750	1595.9700	9.4700	15.0000	17.0000	158.5294	1971.8824
	-	-	-	-	-	-	-
2	137.1000	1550.8537	9.5100	15.0000	18.0000	160.2222	1995.2614
	134.3375	1573.4118	9.4900	15.0000	17.5000	159.3758	1983.5719
	0.0210	-0.0141	0.0021	0.0000	0.0294	0.0053	0.0059
3	118.1550	1516.6340	9.3250	15.0000	17.0000	150.0000	1963.6397
	128.9433	1554.4859	9.4350	15.0000	17.3333	156.2505	1976.9278
	-0.0402	-0.0120	-0.0058	0.0000	-0.0095	-0.0196	-0.0033
4	137.3200	1550.7850	9.1450	15.0000	19.0000	164.7368	1987.4269
	131.0375	1553.5607	9.3625	15.0000	17.7500	158.3721	1979.5526
	0.0162	-0.0006	-0.0077	0.0000	0.0240	0.0136	0.0013
5	135.9750	1562.5000	9.9100	15.0000	23.0000	156.4783	1923.7036
	132.0250	1555.3485	9.4720	15.0000	18.8000	157.9933	1968.3828
	0.0075	0.0012	0.0117	0.0000	0.0592	-0.0024	-0.0056
6	137.9550	1506.9420	9.1050	15.0000	16.0000	162.9375	1971.1250
	133.0133	1547.2808	9.4108	15.0000	18.3333	158.8174	1968.8398
	0.0075	-0.0052	-0.0065	0.0000	-0.0248	0.0052	0.0002
7	138.2950	1567.2277	9.7600	15.0000	19.0000	162.5263	1990.7251
	133.7679	1550.1303	9.4607	15.0000	18.4286	159.3472	1971.9663
	0.0057	0.0018	0.0053	0.0000	0.0052	0.0033	0.0016

Tempo total de execução: 1401 segundos

APÊNDICE B – Exemplos de Mapas Obtidos

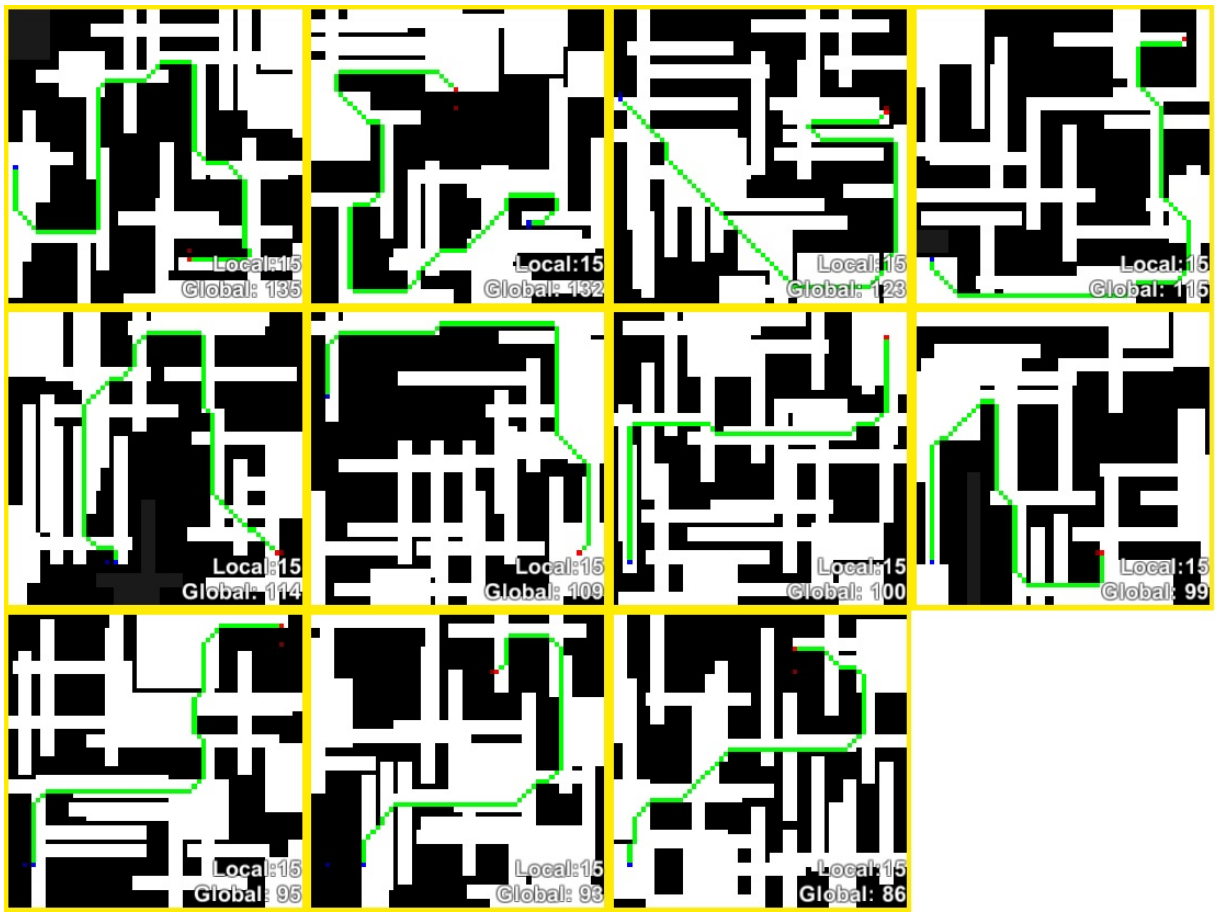
B.1 Algoritmo Genético

Mapas contidos no conjunto C_f da 12ª execução da configuração **AG**.



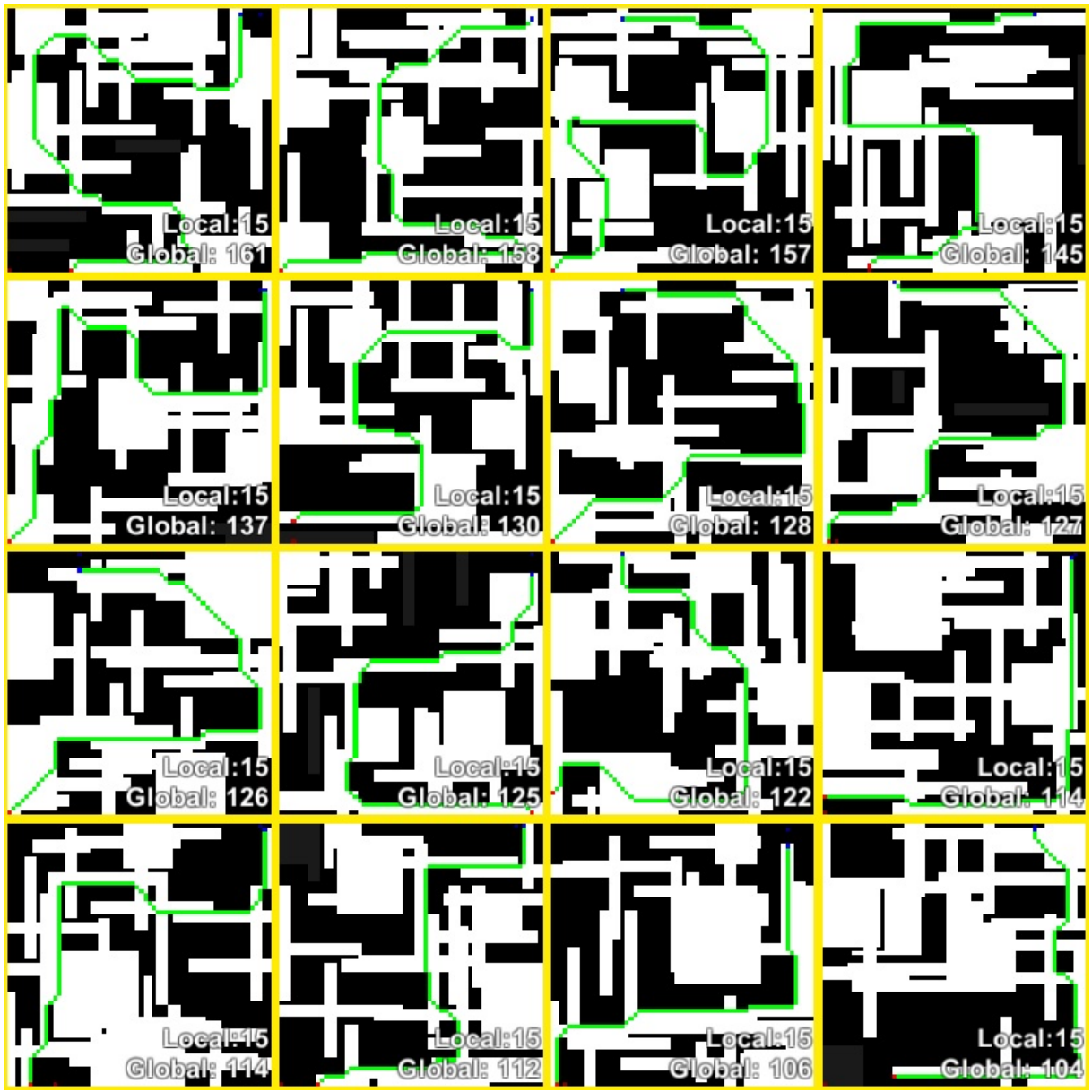
B.2 Busca Inovativa

Mapas contidos no conjunto C_f da 3ª execução da configuração **NS**.



B.3 Competição Local

Mapas contidos no conjunto C_f da 3ª execução da configuração **Lehman**.



B.4 Proposta

Mapas contidos no conjunto C_f da 6ª execução da configuração **Proposta**.