Um Algoritmo Exato para um Problema de Galeria de Arte

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Marcelo Castilho Couto e aprovada pela Banca Examinadora.

Campinas, 13 de Agosto de 2010.

Prof. Dr. Cid Carvalho de Souza (Orientador)

Prof. Dr. Pedro Jussieu de Rezende
(Co-orientador)

Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP
Bibliotecária: Maria Fabiana Bezerra Müller – CRB8 / 6162

Couto, Marcelo Castilho
C837t Um algoritmo exato para um problema de galeria de arte/Marcelo Castilho Couto -- Campinas, [S.P. : s.n.], 2010.

Orientador : Cid Carvalho de Souza ; Pedro Jussieu de Rezende
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1.Geometria combinatória. 2.Otimização combinatória. 3.Programação inteira. 4.Algoritmos. I. Souza, Cid Carvalho de. II. Rezende, Pedro Jussieu de. III. Universidade Estadual de Campinas. Instituto de Computação. IV. Título.

Título em inglês: An exact algorithm for an art gallery problem

Palavras-chave em inglês (Keywords): 1. Combinatorial geometry. 2.Combinatorial optimization. 3. Integer programming. 4. Algorithms.

Área de concentração: Geometria Computacional e Otimização Combinatória

Titulação: Mestre em Ciência da Computação

Banca examinadora: Prof. Dr. Cid Carvalho de Souza (IC – UNICAMP)
Prof. Dr. José Coelho de Pina Júnior (IME – USP)
Prof. Dr. Eduardo Cândido Xavier (IC - UNICAMP)

Data da defesa: 13/08/2010

Programa de Pós-Graduação: Mestrado em Ciência da Computação

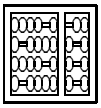
TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 13 de agosto de 2010, pela Banca examinadora composta pelos Professores Doutores:

Prof. Dr. José Coelho de Pina Júnior
IME / USP

Prof. Dr. Eduardo Cândido Xavier
IC / UNICAMP

Prof. Dr. Cid Carvalho de Souza
IC / UNICAMP



Instituto de Computação
Universidade Estadual de Campinas

Um Algoritmo Exato para um Problema de Galeria de Arte

Marcelo Castilho Couto

Agosto de 2010

Banca Examinadora:

- Prof. Dr. Cid Carvalho de Souza (Orientador)
- Prof. Dr. José Coelho de Pina Júnior
  Dep. Ciência da Computação, Universidade de São Paulo (USP)
- Prof. Dr. Eduardo Cândido Xavier
  Instituto de Computação, Universidade Estadual de Campinas (UNICAMP)
- Prof. Dr. Carlos Eduardo Ferreira (suplente)
  Dep. Ciência da Computação, Universidade de São Paulo (USP)
- Prof. Dr. Ricardo Dahab (suplente)
  Instituto de Computação, Universidade Estadual de Campinas (UNICAMP)

Resumo

Nesta dissertação, faz-se um amplo estudo multidisciplinar sobre duas variantes de um problema geométrico NP-DIFÍCIL, o *Problema da Galeria de Arte*, que é analisado tanto pela ótica geométrica quanto combinatória. O objetivo consiste em minimizar o número de guardas suficientes para cobrir todo o interior de uma galeria de arte, representada por um polígono simples. Dentre as muitas variantes desse problema, o foco foi dado àquela onde os guardas são estacionários e restritos aos vértices do polígono, ortogonal ou simples, sem obstáculos.

Propõe-se neste trabalho um algoritmo iterativo exato que é capaz de resolver ambas as variantes do problema. Nesse algoritmo, o problema original é discretizado, reduzido a um problema combinatório, o *Problema da Cobertura de Conjuntos*, e modelado por programação linear inteira. A redução entre os problemas que assegura a corretude do algoritmo e as provas de exatidão e convergência para uma solução ótima do problema original são detalhadas. Apresenta-se também uma extensa experimentação de uma implementação desse algoritmo com o intuito de validar seu uso prático e analisar as várias estratégias propostas aqui para a discretização inicial da galeria.

São dados resultados para instâncias com até 2500 vértices, mais de dez vezes o tamanho das maiores instâncias resolvidas exatamente na literatura pré-existente. Mostra-se que o número de iterações executadas pelo algoritmo está extremamente relacionada com o modo como a galeria é inicialmente discretizada. Considerando a estratégia de discretização com o melhor desempenho geral, tem-se que, na prática, o algoritmo converge para uma solução ótima para o problema original em um baixo tempo computacional e em um número de iterações que é ordens de grandeza aquém do limite teórico resultante da análise de pior caso.

Abstract

In this dissertation, a broad multidisciplinary study is done on two variants of a geometrical NP-HARD problem, the *Art Gallery Problem*, which is approached both from geometrical and combinatorics perspectives. The goal is to minimize the number of guards sufficient to cover the interior of an art gallery whose boundary is represented by a simple polygon. Among the many variants of the problem, the focus was on one where the guards are stationary and are restricted to vertices of the polygon, orthogonal or simple, without holes.

We propose an iterative exact algorithm to solve both variants of the problem. In this algorithm, the original problem is discretized, reduced to a combinatorial problem, the *Set Cover Problem*, and modeled as an integer linear program. The reduction between the problems, which ensures the correctness of the algorithm, and the proofs of its exactness and convergence to an optimal solution are detailed. We also present an extensive experimentation of an implementation of this algorithm in order to validate its practical use and analyze the various strategies proposed here for the initial discretization of the gallery.

Results are given for instances with up to 2500 vertices, more than ten times the size of the largest instances solved to optimality in prior literature. It is shown that the number of iterations performed by the algorithm is highly related to how the gallery is initially discretized. Considering the discretization strategy with the best performance in practice, the algorithm converges to an optimal solution for the original problem in a low computation time and in a number of iterations that is orders of magnitude below the theoretical bound arising from the worst case analysis.

Agradecimentos

A entrega desta dissertação de mestrado é um momento muito feliz para mim. Uma data especial e saudosa, repleta de lembranças boas, compartilhadas com muitos amigos. Esta conquista pessoal deve-se, em grande parte, ao apoio e a presença dessas muitas pessoas que fazem ou fizeram parte, mesmo que por um momento apenas, da minha vida. A elas, eu tenho muito a agradecer.

Em especial, quero registrar meus sinceros agradecimentos aos meus dois orientadores. Eles me acolheram e me orientaram, não apenas academicamente, mas também como um amigo. Não há palavras suficientes para agradecer tudo aquilo que fizeram, em todas as esferas, para que eu concluísse este trabalho. Além de serem profissionais extremamente qualificados e competentes, eles são também pessoas ímpares, as quais eu admiro muito.

Aos meus pais e aos meus tios, agradeço profundamente. Eles sempre estiveram ao meu lado, me incentivando e apoiando em tudo, incondicionalmente. Das minhas primeiras lembranças aos dias atuais, eles sempre estiveram presentes, fazendo o possível para que eu melhorasse sempre. Se hoje me torno Mestre em Ciência da Computação é graças a todo o esforço sobre-humano que eles fizeram.

Agradeço também àquelas pessoas especiais que por nossa vida passam. Desejamos sempre que o tempo delas ao nosso lado não acabe, mas por mais rápida que seja a passagem, trilhamos caminhos juntos e de uma forma boa, elas marcam nossa vida, transmitem alegria e compartilham felicidade. Aprendi e cresci muito com a oportunidade de tê-las por perto. E por isso tenho muito a agradecê-las.

Amizade sincera é um sentimento único, mágico e raro, o qual tenho a alegria de desfrutar. Sou uma pessoa repleta de amigos e muito agradecida por isso. Sempre fui muito feliz nas escolhas dos meus amigos e mesmo distante de alguns, os levo sempre presente em minha vida. Do dia que ingressei no mestrado ao momento da minha defesa, muita coisa boa – e algumas não tão boas – aconteceram, mas durante todas elas, nunca estive só. Sempre houve um amigo por perto para rir, conversar, se divertir, ensinar e até chorar junto. É uma felicidade imensa ter pessoas como vocês para compartilhar os momentos da vida. Agradeço de coração aos meus amigos.

A todos, fica aqui meu muito obrigado! Um grande abraço! Marcelo.

Sumário

Lista de Tabelas

Lista de Figuras

Lista de Algoritmos

Lista de Acrônimos

Capítulo 1

Introdução

Atualmente, a questão da segurança está em destaque na sociedade. Dia após dia, notícias sobre assaltos a museus, bancos, residências e diversos outros locais espalham-se pelos jornais e telejornais do Brasil e do mundo. Com a popularização dos sistemas de segurança, é interessante que haja estudos mais aprofundados para diminuir os custos da implantação e melhorar o aproveitamento desses sistemas.

Os estudos dos chamados *Problemas de Galeria de Arte* vão ao encontro dessa necessidade. Neles analisam-se diversos aspectos inerentes à dificuldade intrínseca dos problemas considerados, a adequação de certas metodologias de resolução, a praticidade dos algoritmos disponíveis, a viabilidade de implantação das soluções propostas e os custos de obtenção de soluções, tanto aproximadas quanto exatas para problemas relacionados a essa questão de segurança. A aplicação desses estudos pode resultar em melhoras sensíveis no desempenho, no aproveitamento e na diminuição dos custos de aquisição desses sistemas.

Dentro desse contexto, nesta dissertação propõe-se um algoritmo exato para duas variantes do Problema de Galeria de Arte. Além da análise teórica da complexidade do algoritmo, é feito um estudo experimental aprofundado sobre o desempenho do algoritmo no qual instâncias de diversas classes são resolvidas à otimalidade.

1.1 Problema de Galeria de Arte

O Problema de Galeria de Arte, proposto em 1973 por Victor Klee [29], consiste em determinar o menor número de guardas – ou câmeras de segurança – suficiente para observar e guardar toda uma galeria de arte. Esse problema pode ser visto como um problema geométrico, onde a galeria é representada por um polígono simples e os guardas por pontos dentro dela. Vasek Chvátal provou em [10] que o número de guardas suficiente, e por vezes necessário, para garantir a cobertura de toda a galeria é o piso de um terço do número de vértices do polígono.

Inúmeras variantes desse problema existem e podem ser obtidas alterando-se determinadas restrições como, por exemplo, o alcance de observação dos guardas, a mobilidade deles, a amplitude do seu ângulo de visão, a existência de obstáculos dentro da galeria e/ou se a galeria é considerada como sendo tri ou bidimensional. No Capítulo 2 discutem-se em detalhes algumas dessas variantes. Para o leitor interessado em aprofundar-se mais sobre esse tema, recomenda-se a leitura do livro clássico de O'Rourke [38] ou ainda dos *surveys* de Shermer [42] e Urrutia [45].

Embora amplamente estudado, poucos são os resultados encontrados na literatura que dizem respeito a algoritmos exatos para o problema de minimização do número de guardas. Por ser um problema NP-DIFÍCIL [34], a maioria dos algoritmos conhecidos resultam em soluções aproximadas que estão a um fator máximo determinado do ótimo. Entretanto, nesta dissertação, é proposto e avaliado experimentalmente um algoritmo exato para duas variantes do problema.

Ambas as variantes tratam do problema de minimização de guardas estacionários nos vértices do polígono que representa a galeria. Nessa situação, pode-se imaginar os guardas como sendo câmeras de segurança cuja localização mais conveniente fica restrita aos vértices do polígono associado à galeria de arte. Além disso, supõe-se que as câmeras possuem um campo de visão ilimitado com 360 graus de amplitude. A primeira variante do Problema de Galeria de Arte que foi pesquisada considera apenas polígonos ortogonais enquanto a segunda trata polígonos simples mas, em ambos os casos, supõe-se que não haja obstáculos no interior da galeria.

Para ilustrar uma aplicação da situação descrita no parágrafo anterior, considere o desenvolvimento de um sistema de segurança usando câmeras para guardar um museu como o Louvre – em Paris, França – ou uma igreja, como a Catedral de Saint Sernin (ver figuras nas páginas 28 e 29) – em Toulouse, França. Pelo método proposto nesta dissertação, inicialmente deve-se desenhar a planta baixa da galeria de arte, de forma que o polígono simples resultante não contenha obstáculos. Por exemplo, na Figura 1.1 vê-se uma imagem aérea correspondente ao museu do Louvre e exibe-se, na Figura 1.2, a sua planta baixa após a remoção dos obstáculos.

Uma vez construído o polígono sobre o qual o problema está definido, o método de resolução faz uma discretização do seu interior em um número reduzido de pontos. Com isso chega-se a uma versão discretizada do problema onde as câmeras devem guardar apenas os pontos constantes da discretização. Continuando com o exemplo do museu do Louvre, na Figura 1.3 os pontos referentes a uma possível discretização do polígono são identificados por uma cruz. É importante observar que existem inúmeras formas diferentes de se fazer essa discretização do polígono. De fato, no Capítulo 3 discorre-se sobre algumas estratégias utilizadas nesse trabalho para essa finalidade.

Uma vez feita a discretização, o algoritmo prossegue resolvendo à otimalidade a

Figura 1.1: Imagem aérea do Museu do Louvre.

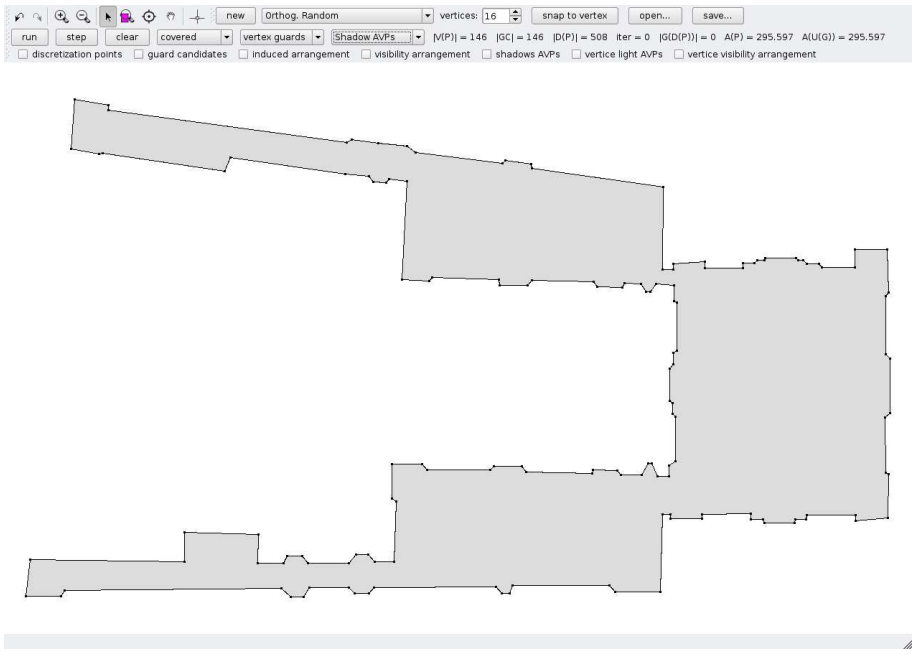


Figura 1.2: Instância da planta baixa sem obstáculos do Museu do Louvre.

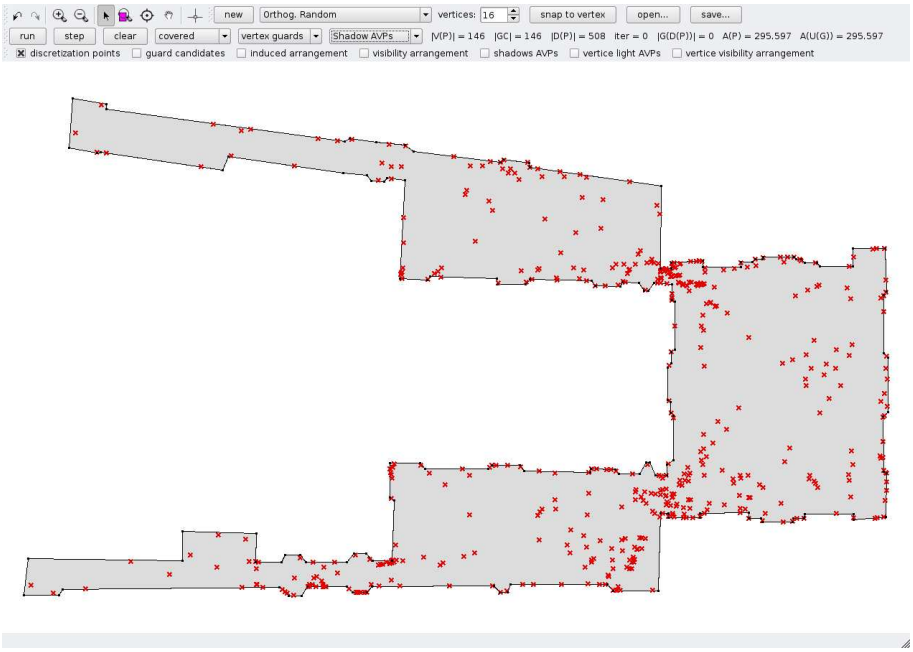


Figura 1.3: Discretização da instância do Museu do Louvre.

instância atual do problema discreto, garantindo-se assim que os pontos da discretização estarão sendo vigiados por câmeras. Contudo, pode ocorrer que as câmeras que otimizam a versão discretizada do problema ainda não sejam suficientes para vigiar todo o polígono, deixando de cobrir regiões onde não existem pontos da discretização. Assim, faz-se um refinamento da discretização incluindo-se pelo menos um ponto de cada uma dessas regiões descobertas, repetindo-se o processo até que não haja mais nenhuma região com essa característica indesejável. Ao final, tem-se então uma solução ótima para o problema original e a posição estacionária dos vértices do polígono onde devem ser colocados câmeras de segurança (ou os guardas). A Figura 1.4 mostra uma solução ótima para a instância baseada na planta baixa sem obstáculos do Museu do Louvre e mostra também a área de cobertura de um dos guardas selecionados. Aspectos relacionados à convergência e à complexidade do algoritmo são discutidos minuciosamente ao longo desse texto.

1.2 Organização da Dissertação

Antes de descrever a organização do texto, a seguinte observação faz-se necessária. Esta dissertação, em concordância com as normas atuais que regem a pós-graduação da Universidade Estadual de Campinas, possui partes do texto escritas em língua estrangeira, correspondentes a artigos publicados ou submetidos a jornais científicos e conferências internacionais. Ainda de acordo com esta norma, o conteúdo desses textos reproduz fi-

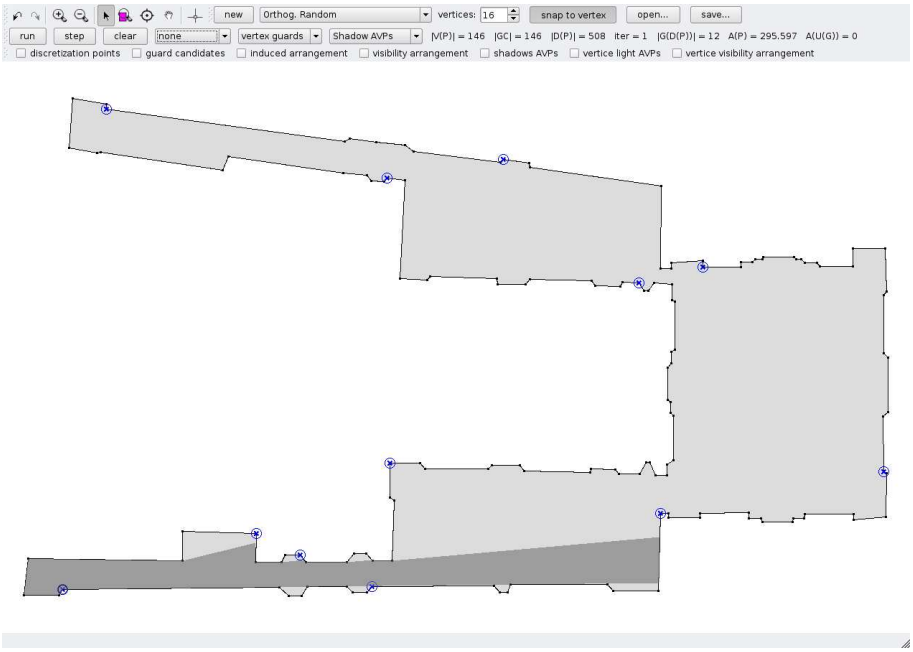


Figura 1.4: Uma solução para a instância do Museu do Louvre e a área de cobertura de um dos guardas.

elmente aqueles dos artigos originais. Contudo, a formatação dos mesmos foi adequada para ficar compatível com o estilo do restante desse documento.

O Capítulo 2 contém os fundamentos teóricos necessários para a compreensão da dissertação. De forma sucinta, inicia com o conceito geométrico do Problema de Galeria de Arte e apresenta uma extensa revisão bibliográfica sobre o tema. Revê também conceitos e definições relevantes para o método proposto, bem como introduz determinadas notações, além de iniciar o leitor na ótica combinatória inerente ao problema. Por fim, mostra os tipos de polígonos utilizados na experimentação da implementação do algoritmo e a regra de geração de cada um.

A abordagem teórica dada ao Problema de Galeria de Arte está reunida no Capítulo 3 que faz uma revisão de todos os pontos importantes do desenvolvimento desta dissertação, desde o início da concepção do algoritmo até a elaboração das estratégias de discretização utilizadas para a experimentação e estudo prático do mesmo. Também descreve a redução do Problema de Galeria de Arte para o Problema de Cobertura de Conjuntos, que retrata muito proximamente como o nosso algoritmo opera.

Os Capítulos 4, 5, 6 e o Apêndice A correspondem a alguns dos artigos publicados ou submetidos a jornais científicos e conferências internacionais. Sua estruturação é feita de forma que haja um prólogo de descrição do artigo, informando os dados de publicação e em qual fase do desenvolvimento da dissertação esse trabalho foi gerado. Na seqüência

tem-se o texto original do artigo, reproduzido fielmente, incluindo sua bibliografia. Às margens do texto, encontram-se alguns apontadores para comentários feitos ao final de cada um desses capítulos. O objetivo desses últimos é complementar o texto do artigo com explicações adicionais que, muitas vezes, foram omitidas desse último por razões de limite de páginas imposto pelo veículo onde se deu a publicação.

As considerações finais sobre a pesquisa desenvolvida nesta dissertação estão apresentadas no Capítulo 7, onde as conclusões são expostas, destacando-se os resultados mais relevantes. Além disso, é dada uma visão geral das contribuições da dissertação e uma lista das publicações obtidas durante o desenvolvimento desse trabalho. Por fim, diversas sugestões são feitas para dar continuidade a este trabalho.

Posto que parte significativa da dissertação corresponde à experimentação da implementação do algoritmo proposto, o Apêndice B retrata o processo de concepção e desenvolvimento dessa implementação. São discutidos nesse apêndice a arquitetura do sistema construído e detalhes da integração com os *softwares* de terceiros que são utilizados como, por exemplo, o resolvedor linear inteiro e a biblioteca de algoritmos geométricos. Além disso, descreve-se como os resultados foram extraídos e quais foram os principais desafios de implementação encontrados. Finalmente, o apêndice apresenta ainda as interfaces criadas para a interação com o usuário e que permitem visualizar facilmente instâncias e soluções do problema tratado, bem como acompanhar a execução do algoritmo passo-a-passo.

1.3 Principais Contribuições

O trabalho realizado nessa dissertação gerou várias publicações entre nacionais e internacionais, algumas das quais aparecem reproduzidas nesse documento. O Capítulo 4 corresponde ao primeiro trabalho, publicado no SIBGRAPI (*Brazilian Symposium on Computer Graphics and Image Processing*) em 2007 e mostra uma versão preliminar do algoritmo proposto e a primeira estratégia de discretização. O segundo trabalho, publicado no WEA (*Workshop on Experimental Algorithms*) em 2008, hoje renomeado como SEA (*International Symposium on Experimental Algorithms*) está descrito no Capítulo 5. Nesse trabalho procurou-se aprimorar o algoritmo através do desenvolvimento de novas estratégias de discretização, além de testar sua robustez executando-o para novas classes de instâncias definidas sobre polígonos mais complexos. Tem-se no Capítulo 6 o último trabalho, publicado como um Relatório Técnico em 2009 no Instituto de Computação da Universidade Estadual de Campinas e submetido a um jornal científico – na data de publicação dessa dissertação o artigo ainda estava em processo de avaliação pelos árbitros. Nesse trabalho, estão as últimas pesquisas realizadas nessa dissertação, incluindo o tratamento para casos de polígonos simples (não ortogonais) e testes exaustivos sobre

estratégias de discretização inicial. O Apêndice A retrata um trabalho intermediário, publicado no GraphiCon (*International Conference on Computer Graphics and Vision*) em 2009.

Há ainda um outro trabalho, publicado no *Symposium on Computational Geometry* em 2009, que é composto de um *resumo estendido* e de um vídeo de aproximadamente seis minutos de duração mostrando o estudo desenvolvido nessa dissertação. Para finalizar, foi criado um *site* na rede mundial de computadores (`www.ic.unicamp.br/~cid/Problem-instances/Art-Gallery`), único do gênero, que contém não apenas todos os dados de entrada das instâncias utilizadas nos trabalhos relacionados acima mas também as soluções ótimas para cada uma delas.

Como a última contribuição tem-se a implementação do algoritmo proposto, com duas interfaces. A primeira, é uma interface de linha de comando e visa auxiliar a automação dos testes. A outra, uma interface visual, tem como objetivo auxiliar a observação de detalhes do problema e do processo iterativo de obtenção da solução.

Capítulo 2

Fundamentação Teórica

Este capítulo aborda conceitos fundamentais relacionados ao Problema de Galeria de Arte (AGP) – do inglês, *Art Gallery Problem*, e ao algoritmo aqui proposto e apresenta as notações e definições mais relevates utilizadas nessa dissertação.

Inicialmente, mostra-se a visão geométrica do AGP explorando alguns conceitos e resultados existentes na literatura, juntamente com as variantes mais estudadas do problema. Em seguida, apresentam-se os conceitos geométricos necessários para compreensão do problema e da solução proposta, seguido da descrição de propriedades combinatórias do mesmo e, por fim, apresenta-se as classes de polígonos utilizadas para validação e experimentação com a regra de formação de cada uma.

2.1 Ótica Geométrica do Problema

O problema da Galeria de Arte clássico foi proposto em 1973 por Victor Klee [29] e consiste em determinar o menor número de guardas suficientes para que todo o interior de uma galeria de arte seja visto por pelo menos um dos guardas. Nesse problema, assume-se que a galeria de arte é composta por n paredes e que cada guarda possui um campo de visão ilimitado de 360°.

A ótica geométrica do problema clássico consiste em representar a galeria de arte como uma região planar delimitada por um polígono simples P, definido por n vértices, que compõem o conjunto V, e por n arestas, de tal forma que nenhum par de arestas não consecutivas possuam um ponto em comum. A denominação "polígono" será utilizada deste ponto em diante para designar um polígono simples e seu interior.

Chvátal [10] estabeleceu o que ficou conhecido como o Teorema da Galeria de Arte de Chvátal: $\lfloor n/3 \rfloor$ guardas fixos em vértices são sempre suficientes e eventualmente necessários para cobrir um polígono P com n vértices. Uma prova alternativa e simples para esse teorema foi apresentada por Fisk [26] em 1978 e baseia-se no fato de que qualquer

triangulação do polígono pode ser 3−colorida.

A Figura 2.1 mostra uma instância da classe de polígonos, denominada Pente$_n$, que estabelece a eventual necessidade de $\lfloor n/3 \rfloor$ guardas.

Prova de Fisk para a suficiência no Teorema da Galeria de Arte de Chvátal.

Inicialmente, triangulariza-se P adicionando $n-3$ diagonais interiores ao polígono. A seguir, colorem-se os vértices de P com três cores distintas de forma que quaisquer dois vértices ligados por uma aresta de P ou por uma diagonal recebam cores diferentes. É fácil notar que V foi particionado em três classes cromáticas distintas (veja Figura 2.1) e que cada triângulo da triangularização possui, necessariamente, um vértice de cada cor. Portanto, pode-se dizer que o conjunto de vértices de qualquer uma das três partições cromáticas guardam o polígono todo, em particular, o conjunto com menor cardinalidade, composto por uma quantidade de vértices menor ou igual a $\lfloor n/3 \rfloor$.

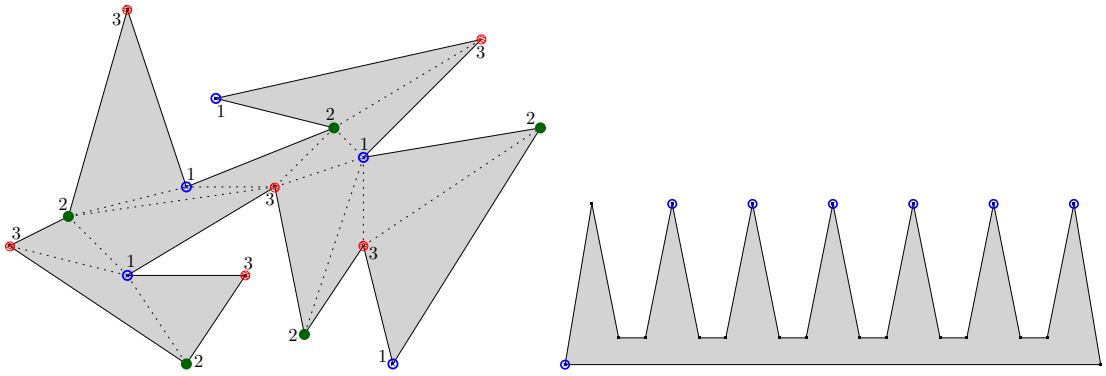


Figura 2.1: (a) Exemplo da prova da suficiência de $\lfloor n/3 \rfloor$ guardas para qualquer instância P: triangulação de P e divisão de V em três classes cromáticas; (b) Classe de polígonos (Pente$_n$) que necessita de $\lfloor n/3 \rfloor$ guardas para ser coberta.

A prova apresentada por Fisk, além de mostrar a suficiência de $\lfloor n/3 \rfloor$ guardas para o problema, ainda pode ser utilizada como um algoritmo para construir efetivamente uma cobertura de qualquer galeria de arte. No entanto, o número de guardas apresentados na solução pode ser muito maior que o necessário em determinadas galerias, como por exemplo, naquelas representadas por polígonos convexos, onde um só guarda é suficiente. Por isso, a busca pelo menor número de guardas para cobrir uma galeria faz-se necessária.

O problema de minimização do número de guardas em vértices para polígonos sem buracos foi provado NP-DIFÍCIL por Lee e Lin [34]. Sua prova, consiste na redução do Problema da 3-Satisfatibilidade Booleana (3SAT) para o AGP na qual a partir de uma instância arbitrária S de 3SAT com m cláusulas e n variáveis booleanas, um polígono simples é construído o qual pode ser coberto com $3m+n+1$ guardas em vértices se e somente se S é satisfatível. A Figura 2.2, extraída de [38], mostra o polígono da redução

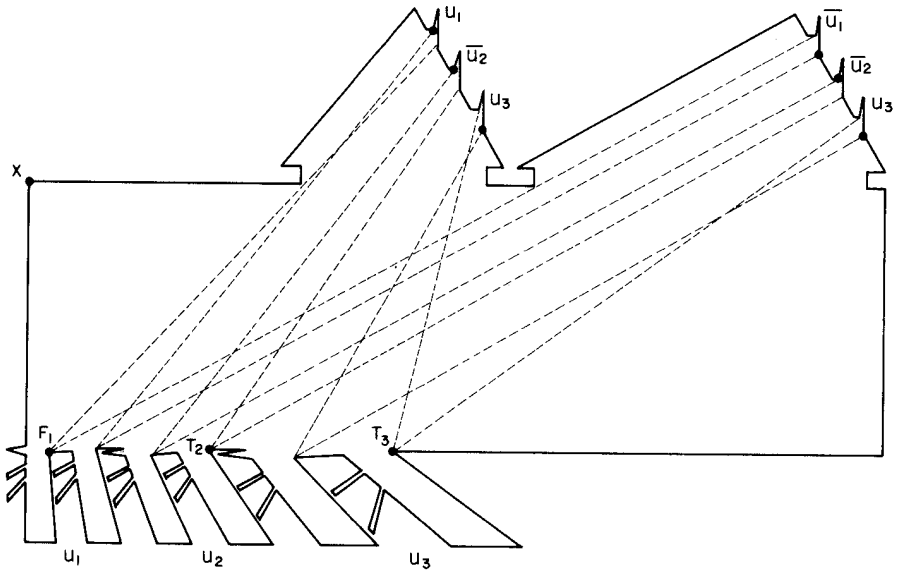


Figura 2.2: Polígono completo da redução de $(u_1 + \overline{u_2} + u_3).(\overline{u_1} + \overline{u_2} + u_3)$ extraído de [38].

de $(u_1 + \overline{u_2} + u_3).(\overline{u_1} + \overline{u_2} + u_3)$. O resultado de NP-dificuldade obtido foi estendido por Aggarwal [5] para guardas em posições arbitrárias.

O problema ortogonal.

Inúmeras outras variantes do Problema de Galeria de Arte existem na literatura e são obtidas ao modificar-se determinado aspecto do problema original. Dentre as variantes existentes, destacam-se aquelas que restringem a forma da galeria a polígonos ortogonais (têm suas arestas paralelas aos eixos x ou y), denominadas Problema de Galeria de Arte Ortogonal (OAGP) – do inglês *Orthogonal Art Gallery Problem*. Visto que, atualmente, a forma ortogonal constitui uma parcela considerável das construções e galerias existentes [45], OAGP é uma importante subclasse do problema original.

Um resultado importante para essa variante do problema foi apresentado por Kahn *et al.* [32]. Eles provaram que $\lfloor n/4 \rfloor$ guardas fixos em vértices são sempre suficientes e eventualmente necessários para se cobrir um polígono ortogonal de n vértices. A prova da suficiência é similar àquela apresentada por Fisk e utiliza-se da quadrilaterização do polígono e uma posterior $4-$coloração dos vértices. A prova da necessidade de $\lfloor n/4 \rfloor$ guardas para determinadas instâncias é sugerida na Figura 2.3, que exibe uma instância de uma classe de polígonos ortogonais derivada do Pente$_n$ e que requer $\lfloor n/4 \rfloor$ guardas.

O problema de minimização do número de guardas para o OAGP foi provado NP-DIFÍCIL somente em 1995, por Schuchardt e Hecker [41], respondendo a uma questão que permaneceu aberta por mais de uma década.

Obstáculos.

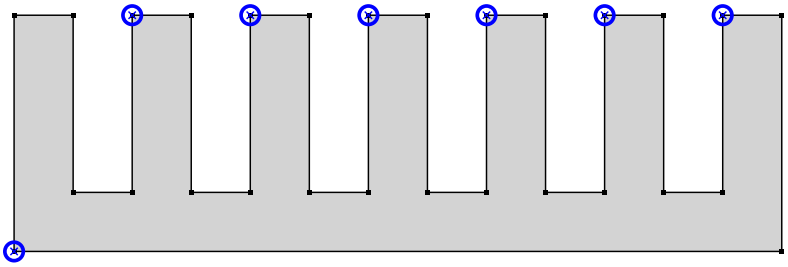


Figura 2.3: Classe de polígonos em que há a necessidade de $\lfloor n/4 \rfloor$ guardas para cobrir polígonos ortogonais.

Pode-se ainda destacar a variante do problema original que lida com obstáculos ou buracos. Nela, os obstáculos são geometricamente representados por polígonos simples disjuntos dentro do polígono P, que corresponde a galeria, de modo que não exista intersecção entre as arestas dos obstáculos e as do polígono. A Figura 2.4 mostra um exemplo de galeria de arte com obstáculos.

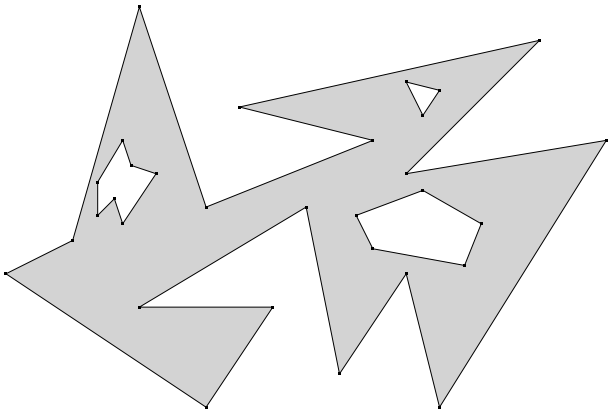


Figura 2.4: Exemplo de uma variante do AGP: galeria de arte contendo obstáculos.

A presença de obstáculos faz com que os limitantes dessa variante sejam diferentes daqueles obtidos para o mesmo tipo de problema com polígonos simples. Os limitantes passam a levar em consideração a quantidade de buracos existentes na galeria.

Outras variantes.

Por fim, a imposição de restrições a uma ou algumas propriedades geométricas, como à localização, à mobilidade e ao poder de visão, seja de ângulo ou de distância, dos guardas utilizados para cobrir uma galeria de arte geram outras variantes do problema original.

Aqui, uma enorme quantidade de combinações de restrições podem ser feitas, tais como, impor que os guardas fiquem fixos e restritos aos vértices do polígono P; permitir que os guardas possam se deslocar, mas somente nas arestas de P; limitar o campo de visão dos guardas a 180° e fixá-los nos vértices da galeria, entre outras.

Essas variantes possuem complexidades diferentes de acordo com a restrição imposta. Em algumas, o problema pode ficar inclusive insolúvel para determinadas instâncias. Abaixo, listam-se as variantes principais encontradas na literatura, no que refere-se às restrições acima consideradas.

Guardas livres. Guardas livres, do termo em inglês *point guards*, é a variante do problema na qual os guardas são irrestritos quanto à localização, isto é, podem estar tanto na fronteira quanto em qualquer ponto dentro do polígono.

Guardas em vértices. Essa variante restringe a localização dos guardas apenas aos vértices do polígono P, tornando finito o conjunto de candidatos a guardas.

Deve-se observar que essa distinção com relação a localização dos guardas é muito importante e pode fazer com que o resultado do problema se altere significativamente. Observa-se na Figura 2.5 um exemplo onde fica claro a diferença de guardas necessários para cobrir uma galeria de arte em cada uma das duas variantes. Em geral, tem-se que os limitantes encontrados para guardas livres e guardas em vértices não são os mesmos.

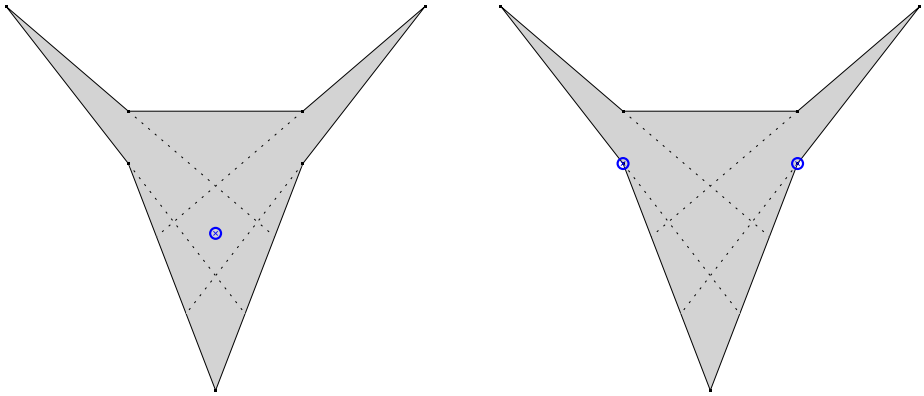


Figura 2.5: Exemplo de polígono que mostra a distinção entre as soluções do problema nas variantes guardas livres e guardas em vértices.

Guardas em arestas. Introduzida em 1981 por Avis e Toussaint [7], a variante guardas em arestas permite que um guarda se mova sobre uma aresta do polígono P.

Um ponto q de P é considerado coberto por um guarda em aresta u se q é visível por u em algum ponto da aresta.

Esse problema pode ser visto sob uma ótica de iluminação, onde cada guarda corresponde a uma lâmpada fluorescente localizada sobre toda a extensão de uma aresta de P. O problema passa a ser o de determinar o menor número de tais lâmpadas a serem acesas para que todo o polígono seja iluminado.

Guardas móveis. Essa variante foi proposta por O'Rourke [37] e permite que os guardas se locomovam sobre segmentos de retas totalmente contidos no interior de polígono.

α-guardas. Proveniente do termo em inglês *α-floodlights*, é a variante do problema

na qual o campo de visão dos guardas é limitado a um ângulo α. Estivill-Castro *et al.* provaram em [24] que essa restrição faz com que o problema seja insolúvel para um determinado conjunto de instâncias e determinados valores de α.

Outros detalhes sobre as variantes descritas encontram-se em *surveys* existentes na literatura, como o publicado por O'Rourke [38], por Shermer [42] e por Urrutia [45].

2.1.1 Variantes Utilizadas na Dissertação

O algoritmo proposto nesta dissertação foi desenvolvido para o problema de minimização de guardas estacionários em vértices do polígono que representa a galeria. Ele pode ser aplicado tanto para polígono ortogonais sem obstáculos (OAGP), quanto para polígonos simples sem obstáculos (AGP).

As diferentes instâncias utilizadas para testes do algoritmo estão descritas em detalhes na Seção 2.4.

2.2 Preliminares Geométricas

Alguns importantes conceitos e definições geométricas são necessários para a compreensão do algoritmo proposto. Esta seção aborda aqueles mais importantes, detalhando quando necessário e unificando a notação.

2.2.1 Região de Visibilidade de um Ponto

A definição de região de visibilidade faz-se necessária para a compreensão do AGP, visto que o conceito de cobrir (isto é, ver) toda uma galeria, é baseado na união das regiões de visibilidade dos guardas do conjunto solução. Na discussão que se segue, a galeria é representada por um polígono simples P.

Dentre os vários parâmetros que influenciam a obtenção da região de visibilidade, os principais são o alcance e o ângulo de visão permitidos. Esses parâmetros diferem nas diversas variações do problema original e assume-se nesta seção que o ângulo de visão permitido é de 360° com alcance ilimitado.

Entende-se por região de visibilidade de um ponto u o conjunto $\mathrm{Vis}(u)$ de todos os pontos pertencentes a P que são visíveis a partir de u. Um ponto q qualquer é visível a partir de u se e somente se o segmento de reta que liga os pontos u e q não intercepta o exterior do polígono P.

Uma outra forma de pensar na delimitação da região de visibilidade de um ponto u é a de imaginar uma luz pontual em u e considerar como $\mathrm{Vis}(u)$ toda a extensão iluminada do polígono.

A Figura 2.6 mostra um exemplo da região de visibilidade de um ponto u. Por sua definição, a região de visibilidade de qualquer ponto, também chamada de polígono de visibilidade, corresponde justamente ao que se denomina, na literatura, um polígono estrelado. Sabe-se que tais polígonos permitem operações mais eficientes do que polígonos simples, como é o caso de localização de pontos em tempo logarítmico no número de seus vértices. Veremos mais tarde que essa característica traz benefícios para a eficiência das computações que descreveremos posteriormente.

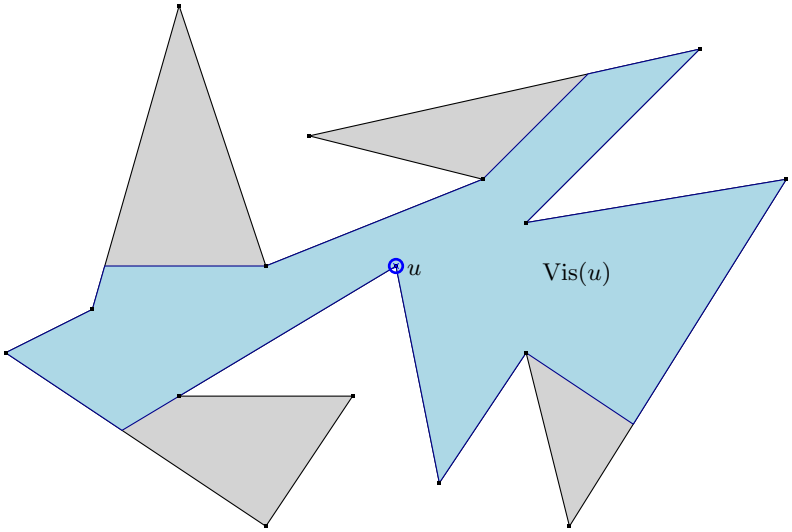


Figura 2.6: Região de visibilidade Vis(u) de um ponto u dentro do polígono P.

Inúmeros algoritmos de diferentes complexidades existem na literatura para a construção da região de visibilidade de um ponto. Em 1983, Lee [33] propôs um dos primeiros algoritmos a possuir complexidade linear para resolver o problema, retificado pouco após por Joe e Simpson [31]. Ainda, Joe e Simpson propuseram em [30] um outro algoritmo linear, parecido com o de Lee, mas que trata casos degenerados, o que invariavelmente acontece ao trabalhar-se com guardas estacionários em vértices e galerias de arte como as descritas na Seção 2.4.

Dada essa natureza das instâncias utilizadas nos experimentos, o algoritmo escolhido e implementado para o cálculo correto das regiões de visibilidade foi o de Joe e Simpson [30]. Durante esse cálculo, o algoritmo faz uso de ângulos para determinar o deslocamento angular (*angular displacement*) e assim computar quais pontos devem fazer parte da região final. Afim de minimizar a ocorrência de erros numéricos faz-se necessária uma técnica de cálculo de ângulos que seja precisa e robusta. A próxima seção descreve tal técnica em detalhes.

2.2.2 Pseudo-ângulos

Para se determinar o ângulo real entre dois vetores de mesma origem, é preciso empregar-se operações não algébricas, cujos resultados dependem da precisão aritmética utilizada e, por isso, estão sujeitos a problemas de aproximação, arredondamento e precisão, os quais se agravam, acumulando-se, quando esses ângulos são utilizados em sequências de cálculos.

No entanto, quando a determinação de ângulos somente é utilizada para fazer comparações entre eles, pode-se aplicar uma técnica que utiliza somente operações algébricas cujos resultados são robustos, precisos e independentes da precisão aritmética utilizada.

Essa técnica, baseada na comparação de pseudo-ângulos [20], consiste em utilizar a avaliação de uma função $f(\theta)$ monotonicamente crescente no ângulo θ para substituir o cálculo explícito dos ângulos sempre que a única operação a ser feita é a comparação entre pares desses valores. A medida de um ângulo θ pode ser vista como o comprimento do arco de uma circunferência unitária delimitado por θ. Correspondentemente, utiliza-se, por exemplo, $f(\theta)$ como o comprimento da porção delimitada por θ do perímetro de um quadrilátero regular simetricamente centrado na origem. A Figura 2.7 ilustra essa forma de cálculo de pseudo-ângulos utilizada para comparações.

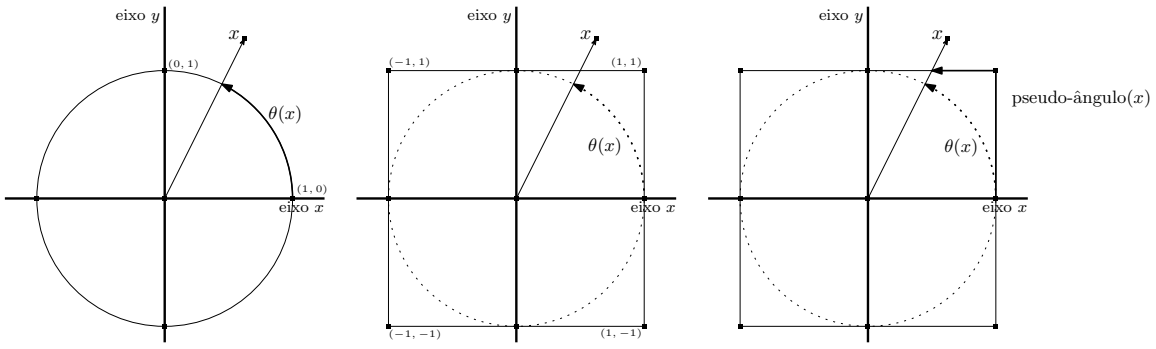


Figura 2.7: Embasamento da técnica de pseudo-ângulos.

Pode-se deduzir facilmente que dadas duas semi-retas x e y, os dois ângulos $\theta(x)$, $\theta(y)$ que elas determinam obedecem à desigualdade $\theta(y) > \theta(x)$ se e somente se $f(\theta(y)) > f(\theta(x))$.

Deste modo, para evitar a necessidade do cálculo explícito de ângulos reais, optou-se pela implementação da técnica de pseudo-ângulos sobre um quadrilátero, centrado na origem, dispensando a computação de funções trigonométricas inversas que induzem a erros de precisão.

2.2.3 Conjunto de Guardas

Considera-se um conjunto de guardas para a galeria de arte representada pelo polígono P o conjunto de pontos G, pertencentes a P, que juntos enxergam toda a galeria.

Portanto, dado um conjunto qualquer de pontos S, candidados a guarda, esse será considerado um conjunto de guardas para P se, para todo ponto $p \in P$ existe um ponto $s \in S$ de tal forma que p seja visto por s.

Tem-se que um conjunto de guardas G é qualquer subconjunto de pontos do polígono P que satisfaz à restrição (2.1).

$$\bigcup_{g \in G} \text{Vis}(g) = P \tag{2.1}$$

Vértice-Guardas.

Um Conjunto de Vértice Guarda (CVG) é definido como qualquer conjunto de guardas que seja um subconjunto dos vértices de P.

Obtem-se portanto, uma ótica alternativa para olhar o Problema de Galeria de Arte, que passa a ser a de encontrar o menor subconjunto $G \subset V$ que seja um CVG para P. Ou seja,

$$\begin{aligned} \text{MIN} \quad & |G| \\ \text{tal que} \quad & \bigcup_{g \in G} \text{Vis}(g) = P, \forall G \subset V \end{aligned}$$

Ao modelar-se o problema da maneira como apresentado acima, chega-se naturalmente a uma forma de resolvê-lo. A próxima seção aborda maiores detalhes sobre as implicações dessa modelagem e o que pode ser feito para otimizá-la.

2.3 Ótica Combinatória

Dada a formulação proposta anteriormente, temos que o problema como um todo passa a ser combinatório. Para resolvê-lo, basta checar todos os possíveis subconjuntos dos vértices para encontrar o menor deles que cubra toda a galeria.

No entanto, conforme o número de vértices cresce, o número de possíveis subconjuntos e, portanto, combinações a serem checadas, cresce exponencialmente, inviabilizando sua aplicação.

Por outro lado, tem-se que, dada a natureza e o tipo do problema, ele pode ser visto como uma versão de minimização *"contínua"* do Problema de Cobertura de Conjuntos (SCP) – do inglês, *Set Cover Problem*. A atribuição *"contínua"* deve-se ao fato que o número de elementos a serem cobertos não é finito.

Problema de Cobertura de Conjuntos.

A modelagem padrão do SCP como um programa linear inteiro consiste de uma entrada composta de um conjunto U constituindo o universo de todos os elementos a serem cobertos e uma família S de subconjuntos de elementos desse universo, cada qual tendo um custo $c(s)$ inerente a sua utilização. Ainda, uma variável x binária é criada para dizer se um determinado conjunto está ou não na solução. Tem-se então a modelagem que segue.

$$\begin{aligned} \text{MIN} \quad & \sum_{s \in S} c(s) x_s \\ \text{tal que} \quad & \sum_{s:\, e \in s} x_s \geq 1, \forall e \in U \\ & x_s \in \{0,1\}, \forall s \in S \end{aligned}$$

Nela, é desejado minimizar a somatória dos custos dos subconjuntos presentes na solução. A primeira restrição garante que todo elemento do universo será coberto por pelo menos um conjunto e a segunda restrição garante que cada conjunto da família apresentada estará ou não integralmente na solução final.

O AGP visto como um SCP.

O Problema de Galeria de Arte pode ser modelado como um Problema de Cobertura de Conjuntos *"contínuo,"* onde o universo a ser coberto é infinito e representado por todos os pontos do polígono P que representa a galeria de arte.

Ainda, o conjunto de candidatos a guarda e suas regiões de visibilidade correspondem a família de conjuntos de elementos do universo a ser coberto. O custo inerente ao uso de cada conjunto é o mesmo, e pode ser definido como 1.

A minimização terá como resultado os guardas e as regiões de visibilidade que fazem parte da solução e que juntos cobrem toda a galeria de arte.

Tem-se portanto que, a princípio, para resolver o Problema de Galeria de Arte, basta resolver o Problema de Cobertura de Conjuntos da forma apresentada.

2.4 Tipos de Polígonos

Visto o caráter prático e experimental da dissertação e a falta de bibliotecas existentes na literatura, com número considerado de instâncias variadas, fez-se necessário a análise e geração de diversos tipos de polígonos para representar as galerias de arte a serem submetidas à implementação do algoritmo proposto.

Cada tipo de polígono construído e utilizado nessa dissertação tem por objetivo atender a uma demanda específica, seja ela a de comparação com resultados prévios existentes na

literatura, seja ela a de explorar e ressaltar determinada condição ou situação pertinente ao algoritmo ou seja ela simplesmente a de ilustrar e complementar a validação almejada.

Ao trabalhar-se com galerias de arte representadas por polígonos e analisar a resposta da implementação do algoritmo às instâncias existentes na literatura, bem como durante estudo e aprofundamento dos conceitos necessários, é natural que sejam propostas evoluções às classes de instâncias existentes, para que o trabalho torne-se completo.

Ao todo, fez-se uso de sete diferentes classes de polígonos para a validação do algoritmo proposto, cada qual possuindo detalhes ou características que a diferencie das demais. Nas subseções seguintes, essas classes são vistas em profundidade, junto com sua regra de formação, exemplos e algoritmos utilizados para a construção das instâncias respectivas.

Quantidade de Instâncias.

Para que o trabalho e os resultados tornem-se estatisticamente confiáveis e as médias apresentadas não distorçam a realidade e a dificuldade do problema, foi feito um estudo prévio para determinar o número mínimo necessário de instâncias a serem geradas para cada valor n de vértices desejado.

Nesse estudo, analisou-se o desvio padrão e a variância dos resultados produzidos pelo algoritmo conforme o número de instâncias com a mesma quantidade de vértices aumentava. A Figura 2.8 mostra, para algumas das estratégias do nosso algoritmo e números de vértices, o desvio padrão do tempo de processamento das instâncias até a otimalidade.

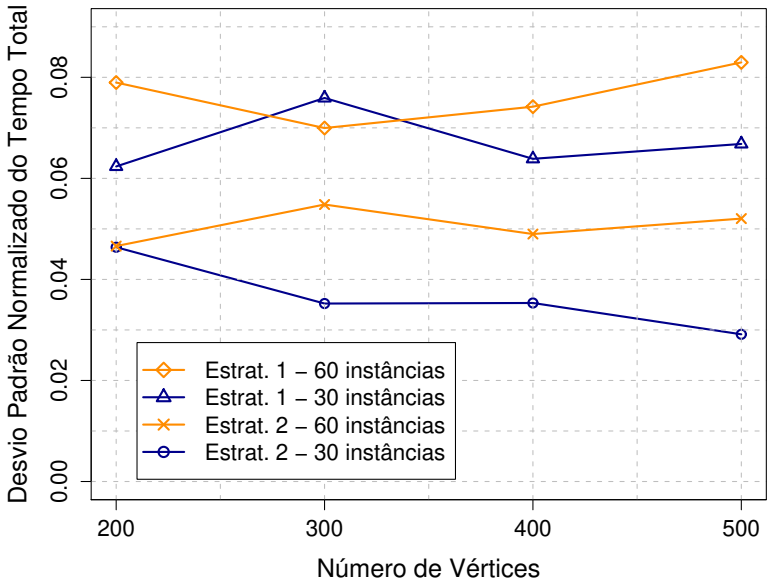


Figura 2.8: Desvio padrão do tempo de processamento para um conjunto de 30 e 60 instâncias aleatórias.

Observou-se que a variância permanece praticamente a mesma quando uma quantidade maior ou igual a 30 instâncias é gerada. Portanto, durante os experimentos, sempre que um resultado for reportado, ele corresponde a média de pelo menos 30 instâncias do mesmo tipo e com a mesma quantidade de vértices.

Portal.

Por fim, para que o trabalho esteja disponível para referências e que seja passível de comparações por trabalhos futuros, criou-se um portal *online* [17] sobre o Problema de Galeria de Arte.

No portal, todas as informações contidas nos artigos publicados estão agrupadas por trabalho e detalhadas, de maneira a facilitar o uso por terceiros. Há, para cada artigo apresentado, todo o conjunto de instância utilizado, a descrição do ambiente computacional em que foi executado os experimentos e todos os resultados obtidos compilados em planilhas CSV, instância à instância. Além disso, ainda há um *link* direto para a página de onde pode ser baixado o artigo.

Cada instância é representada por um arquivo dividido em duas partes. A primeira consiste de um valor inteiro que representa o número de vértices do polígono. A segunda, é uma sequência anti-horária dos vértices, cada qual representado por suas coordenadas x e y, que por sua vez, consistem na divisão de dois números inteiros. Abaixo, temos a representação de um quadrado com coordenadas $(1,1)(50,1)(50,50)$ e $(1,50)$.

$$4\ 1/1\ 1/1\ 100/2\ 1/1\ 500/10\ 50/1\ 1/1\ 100/2$$

Ainda, cada linha da tabela CSV com os resultados obtidos contém a qual instância o resultado pertence, quantos vértices a instância possui, qual foi a estratégia utilizada cujo resultado está reportado na linha, quantas iterações foram necessárias para atingir a solução ótima, qual a quantidade de guardas nessa solução, além do tempo gasto no préprocessamento, processamento e o tempo total necessário para resolver a instância à otimalidade.

2.4.1 *Small Area* ("Polígono de Pequena Área")

A classe de polígonos denominada polígonos de pequena área, foi inicialmente introduzida por Tomás *et al.* em [44] sob o termo MinArea, como um tipo especial de polígonos da classe Thin. Entretanto, da publicação do nosso primeiro trabalho [14] preferiu-se adotar o termo *Small Area* para o mesmo tipo de polígono (veja Pg. 60) devido ao primeiro termo ser considerado pouco adequado.

Esta classe de polígonos é constituída somente de polígonos ortogonais simples que possuem n vértices e não contém arestas colineares, como sugerido por [43]. Ainda, eles são gerados com os vértices sobre uma grade regular unitária de comprimento $n/2 \times n/2$. A Figura 2.9 mostra alguns exemplos de polígonos com $8, 10$ e 12 vértices.

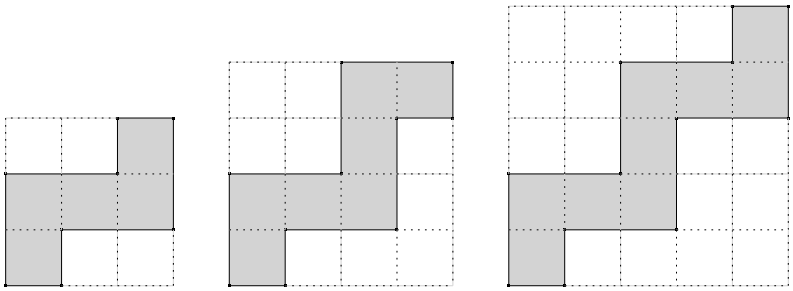


Figura 2.9: Exemplo de polígonos da classe *Small Area*.

Para gerá-los sobre um grid unitário $n/2 \times n/2$, é necessário começar do canto inferior esquerdo com uma célula de uma unidade quadrada e ir caminhando, em diagonal, até o canto superior direito, fazendo com que seja respeitado sempre a propriedade de não haver duas arestas colineares.

Na literatura [44], apresenta-se como um cenário extremo para o problema, devido ao baixo número de restrições presentes no modelo apresentado, o que acarreta a geração de uma matriz muito esparsa para o Problema de Cobertura de Conjuntos. No entanto, depois da experimentação feita em [14], percebeu-se que a classe não possuía a mesma importância para fins práticos e para comparações de resultados, sendo uma classe muito simples para o algoritmo proposto nesta dissertação.

Esta classe faz parte do grupo de testes AGP2007, que possui instâncias com até 200 vértices resolvidas à otimalidade. É interessante notar que, por serem uma família de polígonos ortogonais, o número de vértices dos polígonos dessa classe é sempre um número par.

2.4.2 *FAT* ("Polígono Gordo")

Esta classe, denominada polígonos gordo, foi apresentada, juntamente com os polígonos de pequena área, por Tomás *et al.* em [44] sob o termo FAT. A exceção do primeiro trabalho [14], onde o mesmo tipo de polígono recebeu a denominação *Large Area* para contrapor com as instâncias do *Small Area* (veja Pg. 60), os outros trabalhos adotaram o termo da literatura FAT (veja Pg. 78).

Assim como os polígonos de pequena área, esta classe é constituída apenas por polígonos ortogonais simples, gerados com os vértices sobre uma grade regular unitária de comprimento $n/2 \times n/2$ e não possuem arestas colineares. A Figura 2.10 mostra alguns exemplos de polígonos com $8, 10$ e 12 vértices.

Esta classe possui um método muito similar ao da classe anterior para geração de instâncias, começando pelo canto superior esquerdo da grade regular e indo atingir o canto inferior direito da mesma, respeitando a regra de não possuir arestas colineares e

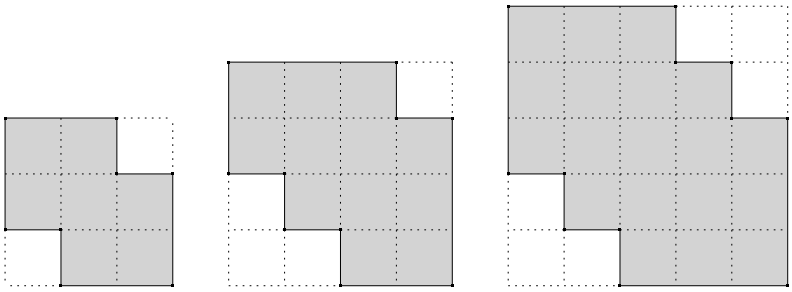


Figura 2.10: Exemplo de polígonos da classe *FAT*.

tentando maximizar a área utilizada. Para isso, basta que todos os vértices da borda inferior do polígono, exceto a ponta, fiquem no quadrante inferior esquerdo e que todos os vértices da borda superior do polígono, exceto a ponta, fiquem no quadrante superior direito, sempre formando uma escada.

Esta classe também foi introduzida como um caso extremo, devido ao grande número de restrições presentes no modelo apresentado, quando da discretização regular. No entanto, para as estratégias mais eficientes desenvolvidas neste trabalho, utilizando propriedades dos polígonos, essa classe torna-se bastante simples, necessitando apenas de uma iteração do algoritmo para provar a otimalidade. Ainda, polígonos *FAT* de qualquer tamanho admitem uma solução analítica ótima de apenas dois guardas, tornando a classe desinteressante do ponto de vista prático.

Esta classe faz parte dos grupos de testes AGP2007 e AGP2008A, que possuem, respectivamente, instâncias com até 200 e até 1000 vértices resolvidas à otimalidade por diversas estratégias diferentes.

2.4.3 Aleatórios Ortogonais

Esta classe, denominada aleatórios ortogonais, é constituída apenas por polígonos ortogonais simples, gerados de maneira aleatória, de acordo com o algoritmo proposto em [43]. Nos trabalhos sobre OAGP, esta classe aparece sobre a denominação *Random* (veja Pg. 60 e Pg. 78). Nos trabalhos mais recentes, foram utilizadas as denominações *Orthogonal Random* (veja Pg. 127) e *Random Orthogonal* (veja Pg. 104) para os mesmos polígonos e assim conseguir distinguí-los da classe dos aleatórios simples, não necessariamente ortogonais.

Assim como em ambas as classes apresentadas anteriormente, as instâncias desta classe também são geradas com os vértices sobre uma grade regular unitária de comprimento $n/2 \times n/2$ e não possuem arestas colineares. A Figura 2.11 mostra um exemplo de polígono desta classe com 200 vértices.

A idéia principal do algoritmo de geração de instâncias dessa classe, proposto em [43], é o de começar por uma célula unitária e, enquanto não chegar ao número de vértices

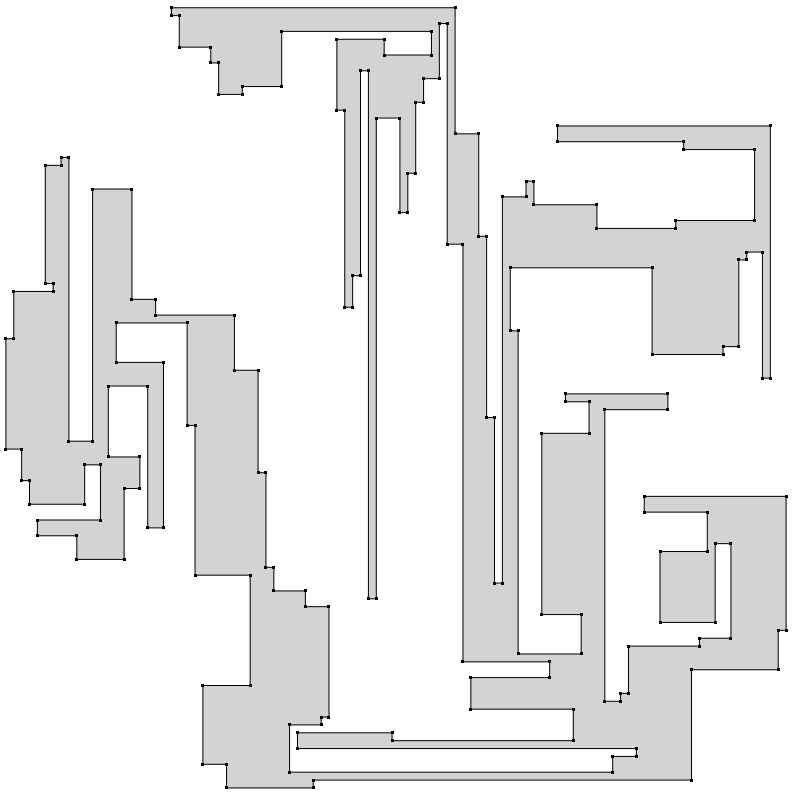


Figura 2.11: Exemplo de polígonos da classe Aleatórios Ortogonais.

desejado, inflar uma célula aleatória do polígono atual e depois recortar um pedaço do polígono aumentado, fazendo com que o número de vértices cresça de dois a cada iteração.

É extremamente importante e útil possuir uma classe com elementos aleatórios do universo de polígonos ortogonais dentre as instâncias de testes. Inicialmente, devido a forma ortogonal constituir uma parcela considerável das construções e galerias existentes na atualidade [45]. No mais, ao analisar-se o comportamento do algoritmo nessas instâncias, tem-se também uma idéia de como será o comportamento médio do mesmo na maioria das vezes que este for aplicado a uma instância ortogonal desconhecida.

Por fim, esta classe faz parte de praticamente todos os grupos de testes (AGP2007, AGP2008A, AGP2008B e AGP2009A), e possui instâncias com até 2500 vértices em alguns deles. Todas as instâncias foram resolvidas à otimalidade pelas diferentes estratégias pesquisadas.

2.4.4 Aleatórios Simples

A classe de polígonos aleatórios simples é formada por todos os tipos de polígonos simples, gerados de maneira aleatória. Utilizada nos trabalhos mais recentes, foi necessária após o

desenvolvimento de estratégias mais robustas, que suportassem sua característica de ser um polígono não ortogonal. Apresentada como *Simple Random* em [16] (veja Pg. 127) foi também denominada *Random Simple* em [11] (veja Pg. 104).

Suas instâncias são geradas distribuindo-se aleatoriamente e uniformemente a quantidade de vértices desejadas dentro de um retângulo de tamanho $0.45n \times n/4$ unidades; após, gera-se um polígono aleatório baseado nesses pontos pelo algoritmo disponível no `CGAL` [1]. Como esse polígono pode não ser simples, ainda é aplicado o método de eliminação de auto-intersecções usando movimentos 2-opt. Pode-se ver um exemplo de polígono desta classe com 200 vértices na Figura 2.12.

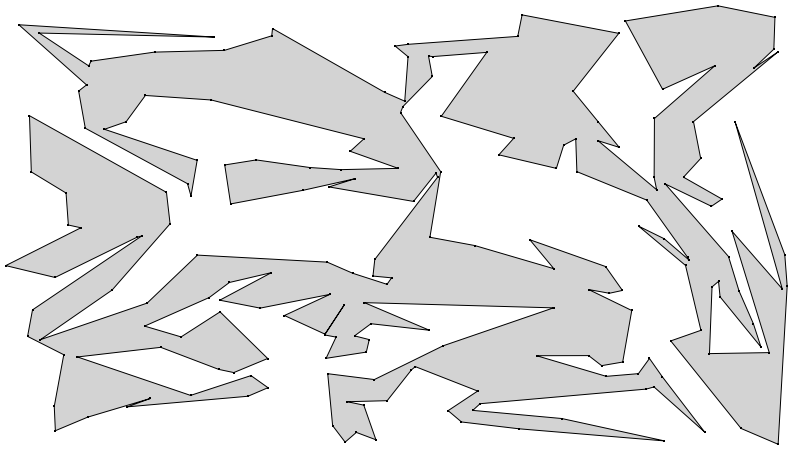


Figura 2.12: Exemplo de polígonos da classe Aleatórios Simples.

É necessário possuir uma classe com elementos aleatórios do universo de polígonos simples dentre as instâncias de testes. Além de ser uma maneira de analisar o comportamento médio esperado do algoritmo para os casos simples, visa também testar o algoritmo, pois, tipicamente, os polígonos obtidos possuem pequenos "dentes" que afloram e impactam no arranjo de visibilidade, em especial nas regiões mais distantes, diferente do comportamento típico dos polígonos aleatórios ortogonais. Ainda, experimentalmente foi observado que os polígonos aleatórios simples são instâncias mais difíceis de serem resolvidas que os aleatórios ortogonais, como pode ser visto em [16, 11].

Esta classe faz parte dos grupos de testes AGP2008B e AGP2009A, que possuem, respectivamente, instâncias com até 1000 e até 2500 vértices resolvidas à otimalidade por diversas estratégias diferentes.

2.4.5 von Koch Completos

A classe dos polígonos von Koch Completos, do inglês *Complete von Koch*, também designada como o acrônimo CvK, é baseada em uma versão modificada da curva de von Koch

(veja [25]). Como todo fractal, ela possui um valor para a dimensão de Hausdorff [28], sendo esse valor 1.34 para a versão aqui apresentada [25].

A geração das instâncias dessa classe de polígonos é feita começando-se por um quadrado e, após, ir recursivamente substituindo cada aresta como mostrado na Figura 2.13, onde $\overline{ar} = \overline{st} = \overline{ub}$ e $\overline{sr} = \overline{tu} = \frac{3\,\overline{ar}}{4}$.

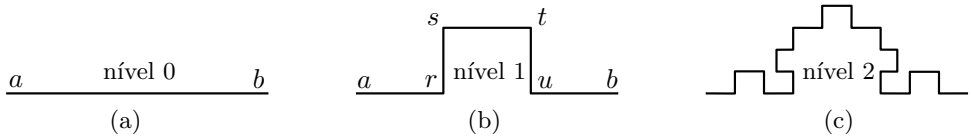


Figura 2.13: Níveis dos polígonos baseados na curva modificada de von Koch.

A Figura 2.13 também introduz uma importante notação que auxiliará na descrição da próxima classe. É dito que, uma aresta do polígono que está sobre a posição do quadrado inicial possui nível 0. Quando ocorre uma operação de substituição, como ilustrado na figura, a uma aresta e do nível k, a todas as arestas que não são colineares a e é atribuído o nível $k+1$. Por exemplo, $\overline{ab}$ (Figura 2.13a) está no nível k, enquanto $\overline{ar}$, $\overline{rs}$, $\overline{st}$, $\overline{tu}$ e $\overline{ub}$ (Figura 2.13b) estão no nível $k+1$

Tem-se que, por construção, os polígonos de von Koch completos possuem um número determinado de vértices. O de nível 0, possui 4 vértices, os de níveis $1, 2, 3$, e 4 possuem, respectivamente, $20, 100, 500$ e 2500 vértices, e assim por diante. A Figura 2.14 mostra a instância de nível 3, com 500 vértices.

A idéia de utilizar um fractal para produzir uma classe de polígonos partiu da busca por instâncias mais complexas, que pudessem perturbar detalhes do algoritmo que possivelmente seriam perturbados apenas por poucas instâncias aleatórias, e mesmo assim, correndo o risco de serem descartados dos resultados por serem considerados *outliers*.

Então, dado o complexo arranjo de visibilidade produzido por esse polígono, ele torna a concorrência entre as estratégias mais leal, realmente selecionando as melhores entre todas, em especial, entre aquelas que fazem uso de estruturas mais complexas e pouco proeminentes nos aleatórios corriqueiros.

Por fim, esta classe faz parte dos grupos de testes AGP2008A, AGP2008B e AGP2009A, e possui instâncias com até 2500 vértices em alguns deles. Todas as instâncias foram resolvidas à otimalidade pelas diferentes estratégias pesquisadas.

2.4.6 Aleatórios de von Koch

Esta classe de polígonos, denominada aleatórios de von Koch, do inglês *Random von Koch*, também referenciada pelo acrônimo RvK, é, assim como a classe anterior, baseada em

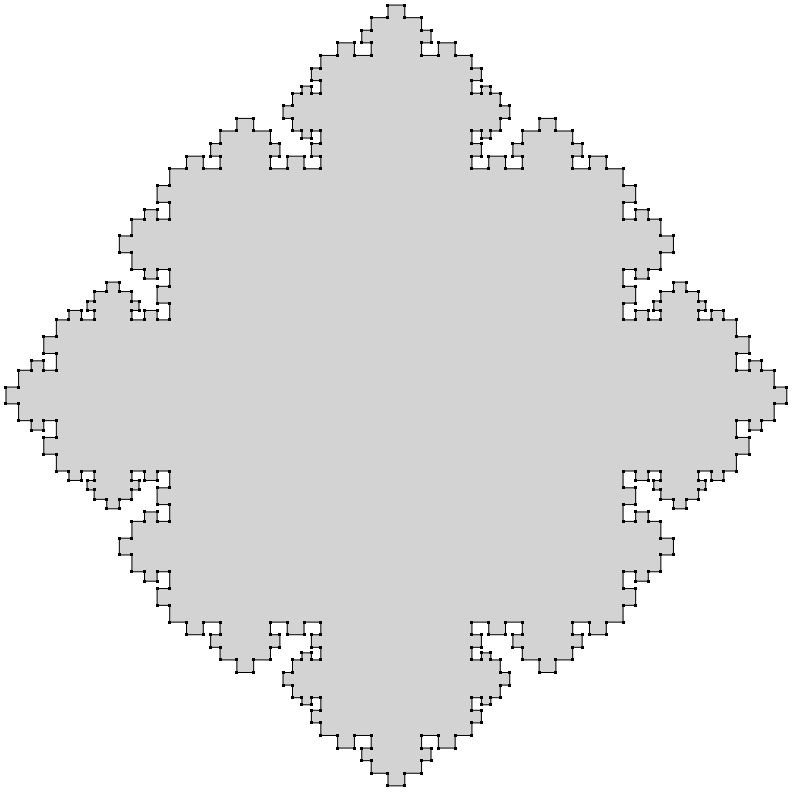


Figura 2.14: Exemplo de polígonos da classe von Koch Completo.

uma versão modificada da curva de von Koch. A diferença entre as classes é o emprego da aleatoriedade para produzir instâncias com a quantidade de vértices desejadas.

Começa-se com um quadrado e aplica-se iterativamente, a uma aresta escolhida aleatoriamente, a operação de substituição mostrada na Figura 2.13. Esse procedimento é feito até que seja atingido o número de vértices desejado. A Figura 2.15 mostra um exemplo de instância dessa classe com 200 vértices.

Detalhadamente, tem-se que em cada iteração, é escolhida uma aresta ao acaso que pertença a um nível inferior a um parâmetro λ determinado, e após, é escolhido também aleatoriamente se essa aresta será ou não expandida com uma probabilidade de 70%. Uma importante observação é que as instâncias de até 1000 vértices foram geradas tendo $\lambda = 4$. Acima desta quantidade de vértices, o parâmetro λ foi setado para 5, visto a proximidade dos 1000 vértices com o polígono completo de von Koch de nível 4, que possui 2500.

Essa é uma explicação para a descontinuidade observada próximo aos 1000 vértices em determinados gráficos de resultados do nosso algoritmo, que podem ser conferidos no trabalho [11].

A importância das instâncias aleatórias de von Koch é a de, devido a pequenas protuberâncias causadas pelos níveis mais elevados, causar pequenas áreas que são vistas

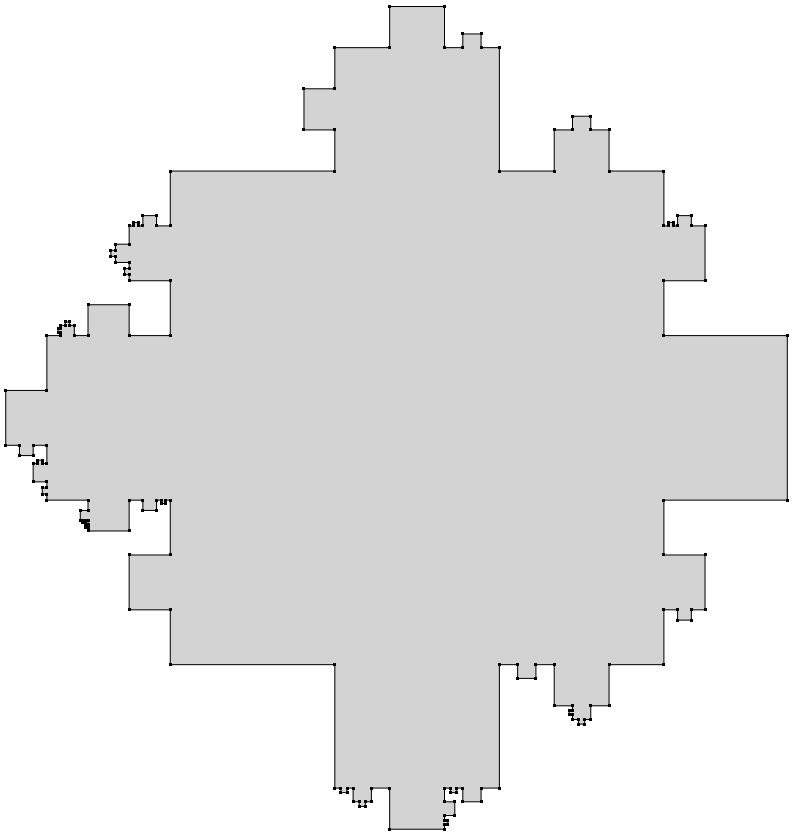


Figura 2.15: Exemplo de polígonos da classe Aleatórios de von Koch.

somente por um pequeno conjunto de vértices, mas que geram perturbações globais no arranjo de visibilidade. Essas instâncias tendem a ser mais complexas e difíceis de serem resolvidas que as aleatórias comuns (veja detalhes na Pg.104).

Por fim, esta classe faz parte dos grupos de testes AGP2008A, e AGP2009A, e possui instâncias com até 2500 vértices. Todas as instâncias ou foram resolvidas à otimalidade pelas diferentes estratégias pesquisadas ou a resolução foi abortada devido à falta de memória da máquina para guardar o arranjo de visibilidade complexo e o modelo do SCP.

2.4.7 Outras Instâncias

Por fim, o algoritmo foi executado para diversas outras instâncias, cada uma visando enfatizar um ponto diferente, ou apenas comparar com uma instância em especial da literatura.

A maioria delas são instâncias produzidas à mão e únicas, não constituindo o volume necessário para serem agrupadas em um conjunto de testes. Um exemplo, é a planificação

feita da Catedral de St. Sernin, que fica em Toulouse na França e pode ser observada na Figura 2.16 e na Figura 2.17.

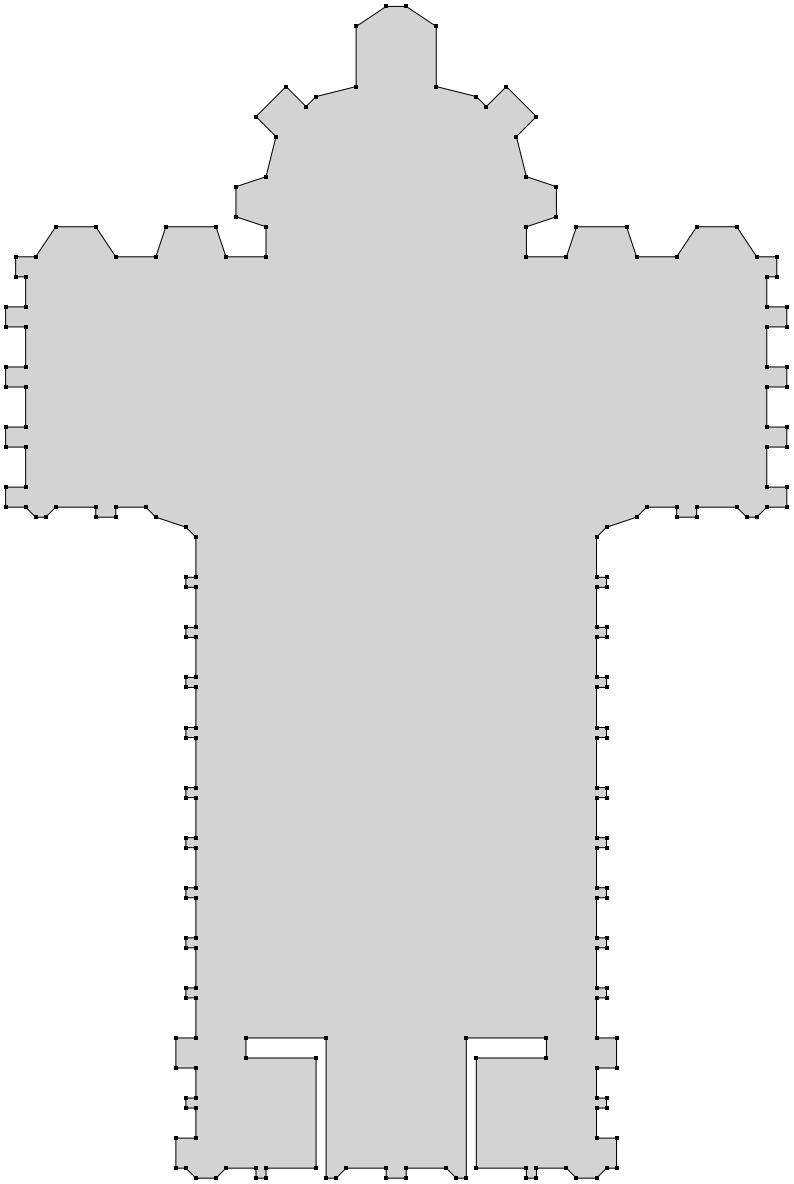


Figura 2.16: Polígono referente a Catedral de St. Sernin, em Toulouse, França.

Figura 2.17: Imagem aérea da Catedral de St. Sernin, em Toulouse, França.

Capítulo 3

Abordagem ao Problema

Este capítulo apresenta a forma como o Problema de Galeria de Arte foi abordado do ponto de vista teórico. Utilizando-se os conceitos, definições e notações apresentados no Capítulo 2, faz-se uma revisão de todos os pontos importantes do desenvolvimento desta dissertação, desde o início da concepção da solução até a elaboração das estratégias utilizadas para a experimentação, avaliação e estudo mais apurado do algoritmo proposto. Também são enunciados e demonstrados os teoremas que dão suporte aos resultados e às decisões tomadas no correr deste trabalho.

Inicialmente, mostra-se como foi feita a concepção do algoritmo e em seguida faz-se um detalhamento maior. Em seguida, apresenta-se uma parte essencial da solução proposta, que é a redução do Problema de Galeria de Arte para o Problema de Cobertura de Conjuntos. Por fim, as estratégias de discretização utilizadas são apresentadas.

3.1 Concepção do Algoritmo

O algoritmo proposto nesta dissertação foi desenvolvido para o problema de minimização de guardas estacionários em vértices do polígono P que representa a galeria. Ele pode ser aplicado tanto para polígonos ortogonais sem obstáculos (OAGP), quanto para polígonos simples sem obstáculos (AGP).

3.1.1 O Problema Contínuo

A abordagem utilizada pelo algoritmo é a de lidar com o Problema de Galeria de Arte utilizando a óptica combinatória de encontrar o menor subconjunto de vértices de P que guarda aquela galeria. Denotaremos um tal conjunto por CVG.

Observa-se na Figura 3.1 um exemplo de uma instância do AGP e a decomposição do polígono nas regiões de visibilidade de cada vértice. Para solucionar o AGP para essa

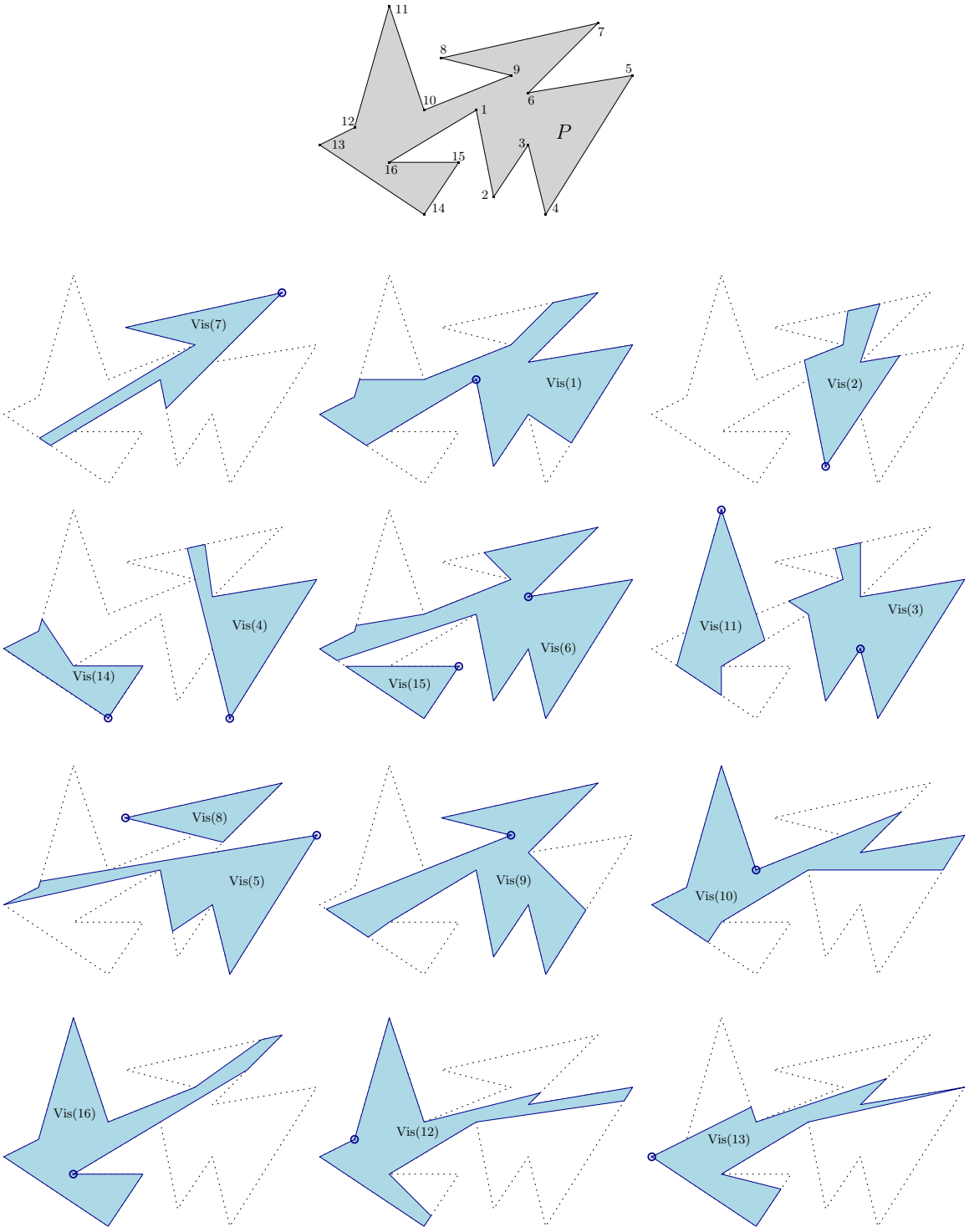


Figura 3.1: Exemplo de polígono e as regiões de visibilidades de cada vértice. Para solucionar o AGP, deseja-se encontrar o menor número de regiões que quando unidas formem o polígono.

instância, é necessário encontrar entre todas as regiões de visibilidades, o menor conjunto que contenha aquelas que, quando unidas, formem novamente o polígono original.

Tem-se então que é matematicamente viável modelar o AGP como um Problema de Cobertura de Conjuntos em um Programa Linear Inteiro (PLI) – veja detalhes na Seção 2.3, onde, o universo a ser coberto é infinito e representado pelos pontos que formam o polígono P, e onde a família de subconjuntos é composta pelas regiões de visibilidade de todos os vértices.

3.1.2 O Problema Discreto

Evidentemente que, embora seja matematicamente viável, é impraticável construir e resolver uma instância do SCP que possua um número infinito de restrições. Portanto, para que esse último problema possa ser usado na resolução do AGP, torna-se necessário utilizar alguma técnica que torne o número de restrições finito e, preferencialmente, pequeno, visto que o SCP é um problema NP-DIFÍCIL.

Para tanto, pode-se discretizar o polígono P que representa a galeria, gerando assim um número finito de pontos $D(P)$ representativos de P, de maneira que o modelo PLI do SCP torne-se computável. De fato, tem-se que nas últimas duas décadas, essa tem sido a única técnica conhecida para desenvolver algoritmos aproximados e eficientes para o Problema de Galeria de Arte [6, 21, 27].

Modelagem do problema discreto como um SCP.

Dada uma discretização arbitrária $D(P)$ do polígono P contendo uma quantidade finita de pontos, a formulação PLI genérica da instância do Problema de Galeria de Arte Discretizado correspondente a um Problema de Cobertura de Conjuntos, denotada por $I(P, D(P))$, é dada por:

$$\begin{aligned} z \;=\; & \text{MIN} \quad \sum_{j \in V} x_j \\ & \text{tal que} \quad \sum_{j \in V} a_{ij} x_j \geq 1, \forall p_i \in D(P) \qquad (1) \\ & \qquad\quad x_j \in \{0,1\}, \forall j \in V, \end{aligned}$$

onde a variável binária x_j assume o valor 1 se e somente se o vértice j de P foi escolhido para integrar o conjunto solução. Ainda, dado um ponto qualquer $p_i \in D(P)$ e um vértice $j \in V$, o valor de a_{ij} é definido como 1 se o ponto p_i é visível por j, ou seja, se $p_i \in \text{Vis}(j)$; caso contrário, o valor de a_{ij} é nulo.

Sendo $D(P)$ uma discretização arbitrária qualquer do polígono P e dado x, uma solução para a instância $I(P, D(P))$ do PLI descrito acima, seja $Z(x) = \{j \in V \mid x_j = 1\}$. Tem-se então que, graças à restrição (1) do modelo apresentado, todo ponto p_i pertencente

à discretização $D(P)$ é visível por pelo menos um guarda presente na solução. Ainda, a função objetivo garante que a cardinalidade z de $Z(x)$ é a menor possível e que a solução satisfaz às restrições, ou seja, garante a cobertura de $D(P)$ por algum dentre os menores conjuntos que a fazem.

Entretanto, é fácil notar que, devido à discretização do polígono possuir um número finito de pontos, não necessariamente a solução $Z(x)$ apresentada garante uma cobertura do polígono P como um todo, como pode ser observado na Figura 3.2. Nela, temos que o conjunto de guardas, embora seja uma cobertura mínima para os pontos da discretização (no caso, todos os vértices de P), não é uma cobertura para P, pois deixa uma pequena região descoberta. Isso faz com que a solução da discretização não seja um CVG para P.

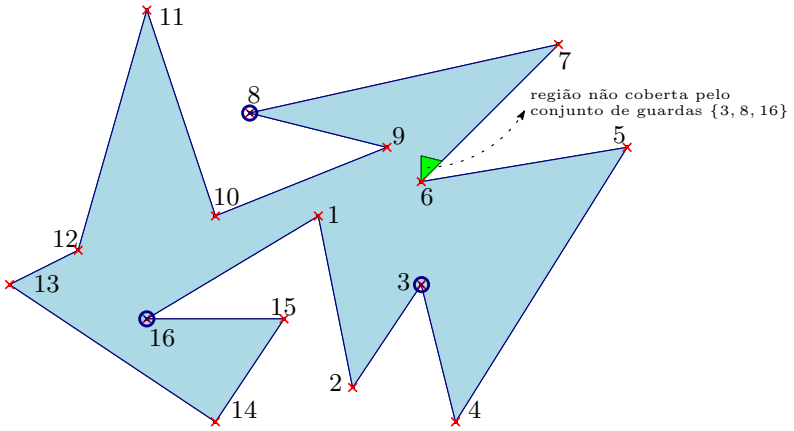


Figura 3.2: Exemplo de instância onde a cobertura de todos os vértices ($D(P)$) não é um CVG para P, pois deixa uma região descoberta.

Qualidade da solução.

É interessante notar que, embora a solução do problema discretizado possa não ser CVG para P, ela ainda pode ser uma boa solução aproximada para o problema original. Sua qualidade pode ser medida, por exemplo, pela porcentagem da área do polígono que não foi coberta, ou ainda pelo número de regiões desconexas não cobertas existentes.

Além disso, mesmo não fazendo parte da investigação feita nesse trabalho, foi observado nos experimentos computacionais realizados que a mudança na estratégia de discretização inicial de P faz com que as regiões não cobertas variem em tamanho e em localização dentro do polígono, por vezes refletindo o potencial de dificuldade de serem vistas por guardas localizados em vértices.

3.1.3 Do Discreto ao Contínuo

Como visto anteriormente, ao construir $D(P)$ como uma discretização arbitrária qualquer de P, não se pode garantir que uma solução ótima para o problema discreto seja sequer factível para o problema contínuo. Isso faz com que qualquer método que pretenda resolver à otimalidade o AGP e que utilize a discretização do polígono P como ferramenta, tenha que lidar de alguma forma com a possibilidade de uma solução do problema discreto não ser uma solução do problema contínuo.

Ao identificar essa situação, pode-se gerar e agregar um novo ponto à discretização de P e, então, construir uma nova instância do SCP e resolver a nova formulação PLI. Esse processo deve ser repetido até que a solução do problema discreto torne-se viável para o problema contínuo. É fácil ver que, nesse último caso, a solução também é ótima para o problema contínuo. Porém, legitimamente, pode-se perguntar se esses passos podem ficar se repetindo indefinidamente sem que essa situação ideal seja atingida. Contudo, prova-se na Seção 3.2.3 que o processo sempre converge para uma solução do problema contínuo em um número finito de passos.

Incidentalmente, convém citar que uma abordagem baseada em programação linear para estimar limitantes para o número de guardas necessários para monitorar uma galeria de arte foi tentada em [8], muito embora nenhuma garantia de convergência ou de obtenção de uma solução exata tenha sido ali apresentada.

Note-se que o número de passos do procedimento descrito acima depende da geração de regiões não cobertas e que a cada iteração um modelo PLI de um problema NP-DIFÍCIL é resolvido, cuja quantidade de restrições corresponde ao número de pontos da discretização utilizada. Em virtude disso, uma investigação interessante diz respeito à variação da estratégia utilizada para gerar o conjunto $D(P)$ inicial. Vale observar que qualquer método que gere uma discretização inteligente incorre em um custo de preprocessamento associado à obtenção dessa estrutura, o qual deve se contrapor àquele despendido com a resolução do problema NP-DIFÍCIL.

Nesse contexto, a Seção 3.4 mostra em detalhes as diferentes estratégias de discretização investigadas nos trabalhos publicados [16, 11, 14] e reproduzidos nos capítulos subseqüentes, onde pode ser vista a evolução que acompanhou a geração das estratégias e uma profunda investigação sobre os aspectos positivos e negativos de cada uma delas.

Viabilidade da solução discreta.

Sabe-se que uma solução para o AGP discretizado será também uma solução para o problema contínuo quando essa não deixa regiões do polígono P sem cobertura. Formalmente, quando isso ocorre, diz-se que a solução é viável.

Seja $I(P, D(P))$ uma instância do Problema de Galeria de Arte Discretizado com o polígono P representando a galeria e $D(P)$ uma discretização arbitrária de P. Considere uma solução x do PLI correspondente a essa instância e seja $Z(x) = \{j \in V : x_j = 1\}$,

ou seja, $Z(x)$ é o conjunto de vértices de P cujas variáveis associadas tem valor um na solução x. Nesse caso, diz-se que $Z(x)$ é *viável* se $Z(x)$ for um CVG para P, i.e., se

$$\bigcup_{g \in Z(x)} \text{Vis}(g) = P.$$

Otimalidade de uma solução viável.

Como mencionado anteriormente, ao identificar que uma solução $Z(x)$ não é viável, pode-se agregar à discretização $D(P)$ um novo ponto de P não coberto por essa solução e repetir esses passos até que o processo convirja para uma solução viável para o problema contínuo. O teorema que se segue prova que a solução obtida para a última discretização gerada por esse processo iterativo é, de fato, uma solução ótima para o AGP original.

Teorema 3.1.1. *Seja Z uma solução ótima para uma instância $I(P, D(P))$ do Problema de Galeria de Arte Discretizado. Se Z é viável, então Z é ótima para o problema contínuo original definido sobre essa mesma instância.*

Prova. Como Z é uma solução ótima para o problema de minimização definido para $I(P, D(P))$, evidentemente, Z deve ser um conjunto de vértices que guarda o conjunto de pontos $D(P)$ que discretiza o polígono P. Ainda, $z = |Z|$ é a menor dentre as cardinalidades de todos os conjuntos de vértices que guardam $D(P)$.

Seja Z^* uma solução ótima para o problema contínuo e seja $z^* = |Z^*|$. Portanto, Z^* deve ser um CVG para P e z^* a menor dentre as cardinalidades de todos os conjuntos que são CVGs para P.

Como Z^* também é uma solução que cobre os pontos da discretização $D(P)$, ela é um conjunto de vértices que guarda $D(P)$ e, portanto, é necessário que $z^* \geq z$.

Por outro lado, tem-se que, como Z é viável, ele também é um CVG para P e, portanto, é necessário que $z \geq z^*$. Isso conclui a demonstração. □

Com isso, fica mostrado que quando o método iterativo encontra uma solução para o Problema de Galeria de Arte Discretizado que seja viável, essa solução também é mínima e é um CVG para P, ou seja, também é uma solução para o AGP original contínuo. Mais adiante na Subseção 3.2.3 será provado que a convergência para uma solução ótima do AGP é alcançada em um número finito de passos.

3.2 Algoritmo

O algoritmo, inicialmente proposto no trabalho [14], leva em consideração todos os pontos observados na seção anterior. Ele foi concebido com duas fases distintas, cada uma responsável por uma parte essencial para a obtenção do resultado final esperado.

A primeira fase é constituída de um preprocessamento, onde uma das estratégias para gerar a discretização inicial, descrita em detalhe na Seção 3.4, é utilizada e a formulação do PLI é construída.

A segunda fase do algoritmo tem como entrada a discretização inicial e a sua formulação PLI. Como foi descrito anteriormente e será detalhado a seguir, nessa fase executa-se um processo iterativo que resolve a instância do SCP para a discretização corrente, até que não exista mais nenhum ponto descoberto do polígono P e, quando esse não for o caso, atualiza adequadamente a discretização e o modelo PLI correntes.

A seguir, as duas fases do algoritmo são descritas em detalhes, inclusive com seus pseudocódigos.

3.2.1 Preprocessamento

Durante a fase de preprocessamento, todas as estruturas necessárias bem como os objetos que serão reutilizados durante a próxima fase são gerados. Ao todos, três importantes passos são executados: o primeiro computa todos as regiões de visibilidade para os pontos $u \in V$; o segundo, computa a discretização $D(P)$ inicial; e o último constrói o modelo PLI correspondente aos dados da entrada e à discretização inicial.

O Algoritmo 3.2.1 sumariza os principais passos da fase de preprocessamento. Para construir a formulação PLI descrita nas seções anteriores, começa-se gerando a discretização inicial $D(P)$ do polígono P (passo 1). Na Seção 3.4 são discutidas diferentes alternativas para gerar essa discretização e o impacto delas na eficiência do algoritmo.

Uma vez que a discretização inicial está construída, são computados quais dos seus pontos estão localizados dentro de cada região de visibilidade de todo vértice $u \in V$, e essas restrições são então incluídas na formulação PLI do SCP.

Algorithm 3.2.1 Preprocessamento

1: $D(P) \leftarrow$ construção da discretização inicial de P;
2: **for** cada $j \in V$ **do**
3: computar $\text{Vis}(j)$;
4: **for** cada ponto da discretização $p_i \in D(P)$ **do**
5: $a_{ij} \leftarrow \textbf{boolean}(p_i \in \text{Vis}(j))$;
6: **end for**
7: **end for**

Repare que, como a complexidade de computar a região de visibilidade é $O(n)$ [30] (ver Seção 2.2.1), a complexidade total do passo 3 do algortimo é $O(n^2)$. Assumindo que $m = |D(P)|$, tem-se que a complexidade de dizer se cada um dos pontos da discretização $D(P)$ está localizado dentro ou fora de cada polígono estrelado que representa a região de visibilidade de cada vértice é $O(\log n)$. Como existem n regiões de visibilidade e m pontos na discretização, a complexidade total do passo 5 é $O(nm \log n)$.

Observe que a complexidade total da fase de preprocessamento é dada pelo passo 5 sempre que $m \in \Omega(n/\log n)$. Caso contrário, a complexidade é dominada pelo passo 3.

Por fim, tem-se que o resultado da fase de preprocessamento é uma formulação de um Programa Linear Inteiro do Problema de Cobertura de Conjuntos que, ao ser resolvido, gera uma solução Z que, embora não necessariamente constitua um CVG para P, garante que todos os pontos da discretização $D(P)$ serão cobertos.

3.2.2 Processamento

A segunda fase do algoritmo computa a solução do Problema de Galeria de Arte original. Para isso, o PLI do SCP correspondente ao problema discretizado é resolvido e refinado, adicionando-se novas restrições referentes às áreas não cobertas, até que uma solução viável (e portanto, ótima) seja encontrada.

Esse refinamento é feito gerando um ponto a mais para a discretização de P para cada região não coberta pela solução atual. Esse ponto, por exemplo, o centróide de cada região descoberta, é então adicionado ao modelo PLI existente por meio de uma nova restrição ao conjunto de restrições (1). Isso garante que, na próxima iteração, aquele ponto em especial e uma boa parte daquela região será coberta, tornando a discretização de P mais densa e aproximando-se, assim, de uma solução viável.

O Algoritmo 3.2.2 mostra em detalhes os passos executados pela última fase do método, o processamento. Repare que o número de iterações depende da quantidade e do momento em que as regiões não cobertas são encontradas. Ainda, será provado na próxima seção que o método converge sempre em um número de passos que é limitado polinomialmente no número de vértices do polígono P.

Algorithm 3.2.2 Processamento

1: **repeat**
2: $Z \leftarrow$ solução de $I(P, D(P))$;
3: **for** cada região não coberta R **do**
4: $c \leftarrow$ centróide de R;
5: $D(P) \leftarrow D(P) \cup \{c\}$;
6: Adicione uma nova linha, r, ao conjunto de restrições (1) da formulação PLI, correspondente ao novo ponto c:
$\sum_{j \in V} a_{rj} x_j \geq 1$ onde $a_{rj} \leftarrow$ **boolean**$(c \in \text{Vis}(j))$, $\forall j \in V$;
7: **end for**
8: **until** Z seja viável

3.2.3 Convergência

Dada a descrição do algoritmo e a prova do Teorema 3.1.1, que estabelece que, uma vez encontrada uma solução viável para o problema discretizado, ela será também ótima para o problema contínuo, resta ser provado que o Algoritmo 3.2.2 converge em um número finito de passos.

Para provar a convergência, é necessário introduzir as definições de *arranjo de visibilidade* e de um Polígono Atômico de Visibilidade (AVP) – do inglês, *Atomic Visibility Polygon*.

Arranjo de visibilidade e AVP.

Considere o conjunto de todas as regiões de visibilidade dos vértice de P, cuja união, obviamente, cobre P. Define-se como arranjo de visibilidade o arranjo de segmentos de retas induzido pelas arestas dessas regiões dentro do polígono P. Ainda, tem-se que cada uma das faces desse arranjo é denominada um Polígono Atômico de Visibilidade, ou AVP. A Figura 3.3 mostra um exemplo de um polígono e seu arranjo de visibilidade com suas faces (AVPs).

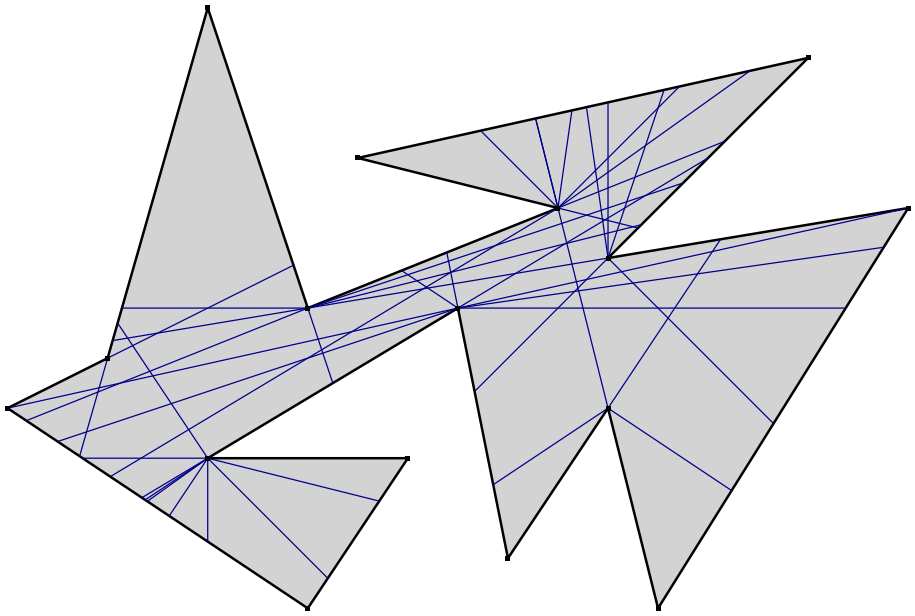


Figura 3.3: Exemplo de arranjo de visibilidade e suas faces, ou AVPs.

Como a região de visibilidade de qualquer vértice tem, no máximo, $O(n)$ arestas, o arranjo induzido por essas regiões será gerado por $O(n^2)$ segmentos de retas, o que garante que sua complexidade total será de no máximo $O(n^4)$ faces (ou AVPs). Na verdade, um resultado de Bose *et al.* [9] mostra que $\Theta(n^3)$ é um limitante ainda mais justo para o número de AVPs.

É importante observar que, da definição de AVPs apresentada, tem-se que quando o centróide de – ou na realidade, quando qualquer ponto no interior de – um Polígono Atômico de Visibilidade $\mathcal{V}$ é visto por um guarda estacionário em um dos vértices de P, toda a face $\mathcal{V}$ também é, obrigatoriamente, vista por esse mesmo guarda. Observe que, se esse fato não fosse verdadeiro, a região $\mathcal{V}$ deveria, necessariamente, ser cortada por alguma aresta da região de visibilidade do vértice em questão, o que contradiz a hipótese de que $\mathcal{V}$ seja um AVP.

Implicações para convergência do algoritmo.

Observe que no passo 3 do Algoritmo 3.2.2, qualquer região não coberta e que testemunhe o fato que Z não seja viável, é, necessariamente, um polígono simples formado pela união de AVPs vizinhos, ou seja,

$$R_i = \bigcup_{j=1}^{k} \text{AVP}_j, \text{ para algum } k.$$

Portanto, um limitante superior para o número máximo de iterações efetuada pelo algoritmo é o número de AVPs possíveis, visto que em cada iteração será gerado, na pior das hipóteses, um ponto em apenas um dos AVPs do arranjo de visibilidade de P. Tem-se então, a prova necessária de que o algoritmo converge, no pior caso, em $\Theta(n^3)$ iterações.

Como será visto na discussão dos resultados computacionais alcançados nessa dissertação, na prática [11], o algoritmo apresentado converge em um número de iterações muito inferior a esse limite teórico de $\Theta(n^3)$. Contudo, duas observações devem ser ressaltadas aqui. Primeiro deve-se estar atento ao fato de que, a cada iteração, uma instância de um problema NP-DIFÍCIL, o SCP, é resolvida, podendo levar um tempo exponencial. Além disso, deve-se perceber que se for utilizada uma discretização inicial formada por exatamente um ponto de cada AVP, o algoritmo irá convergir para a solução ótima do AGP em apenas uma iteração. Como se verá adiante, essa última situação será explorada em nossos experimentos.

3.3 Redução: AGP $\preceq_P$ SCP

Uma importante parte do trabalho, baseada nos resultados descritos e detalhados acima, é a redução polinomial do Problema de Galeria de Arte para o Problema de Cobertura de Conjuntos. Discutiremos inicialmente uma redução de Turing. Discussões sobre as implicações decorrentes do uso de diferentes formas de reduções e suas implicações na Teoria da Complexidade, ainda que relevantes, estão muito além do escopo deste trabalho. Para o leitor interessado no relacionamento entre os diferentes tipos de reduções e a definição das classes de problemas, recomenda-se a leitura do exemplo discutido em [39],

páginas 397–398, onde aplica-se uma redução da forma daquela que é feita a seguir.

A idéia principal é a de, inicialmente, transformar a instância do problema contínuo $I_{AGP}(P)$ para o problema discreto $I_{AGP}(P, D(P))$ e então modelá-la como uma instância do SCP $I_{SCP}(P, D(P))$, que deve ser resolvida. Tem-se então a transformação da solução do SCP na solução para o problema discreto do AGP. Após, verifica-se se essa solução é viável e caso não seja, refina-se a discretização, gerando uma nova instância para o problema discreto e uma nova iteração. Os passos dessa redução iterativa são destacados e detalhados a seguir.

1. $\boldsymbol{I_{AGP}(P) \xrightarrow{\propto_P} I_{AGP}(P, D(P))}$

Dada uma instância qualquer do AGP contínuo $I_{AGP}(P)$, onde P é o polígono que representada a galeria, basta construir uma discretização arbitrária $D(P)$ para P e resolver a mesma instância para o AGP discreto, $I_{AGP}(P, D(P))$.

2. $\boldsymbol{I_{AGP}(P, D(P)) \xrightarrow{\propto_P} I_{SCP}(P, D(P))}$

Assume-se como conjunto universo os pontos da discretização $D(P)$ e como família de conjuntos cada uma das regiões de visibilidade dos vértices, onde cada membro do conjunto universo é atendido pelo conjunto u se e somente se o ponto está dentro da região de visibilidade do vértice correspondente a u. Por fim, assume-se que o custo de utilizar cada membro da família de conjuntos é 1.

3. Resolver $\boldsymbol{I_{SCP}(P, D(P))}$

Para resolver a instância gerada do SCP, utiliza-se um resolvedor de PLI[1].

4. $\mathbf{Res}(\boldsymbol{I_{AGP}(P, D(P))}) \xleftarrow{\propto_1} \mathbf{Res}(\boldsymbol{I_{SCP}(P, D(P))})$

Basta colocar na solução do AGP discreto o guarda referente aos conjuntos escolhidos no SCP.

5. $\bigcup_{g \in G} \mathbf{Vis}(g) = P$ **?**

Verificar se a união das regiões de visibilidade de cada vértice da solução é o polígono P.

5.1 (não) $\boldsymbol{D(P) \leftarrow D(P) \cup \{c(R_i)\}}$

Caso não seja, deve-se então refinar $D(P)$, adicionando-se à discretização o centróide de cada uma das regiões não cobertas R_i e, feito isso, ir novamente para o passo 2 da redução.

5.2 (sim) $\mathbf{Res}(\boldsymbol{I_{AGP}(P)}) \xleftarrow{\propto_1} \mathbf{Res}(\boldsymbol{I_{AGP}(P, D(P))})$

[1]Lembra-se que este passo pode levar tempo exponencial devido ao problema ser NP-DIFÍCIL.

Uma vez que a solução para a instância do problema discreto é viável, tem-se que ela pode ser utilizada como uma solução para a instância do problema contínuo.

Por fim, a Figura 3.4 esquematiza todos os passos apresentados dessa redução (de Turing).

Cabe aqui um breve comentário. As reduções de Karp, usadas em provas de NP-completude, não permitem este tipo de redução iterativa entre problemas. Entretanto, optamos por descrever acima uma redução de Turing já que ela retrata muito proximamente como o nosso algoritmo opera.

Contudo, o leitor atento deve ter se apercebido que, como decorrência das observações sobre convergência feitas na seção 3.2.3, bastaria que fizéssemos a discretização $D(P)$ da transformação $I_{AGP}(P) \xrightarrow{\propto_P} I_{AGP}(P, D(P))$ ser composta dos centróides de todas as AVPs do arranjo de visibilidade de P, para que uma única iteração do passo **3.** acima fosse suficiente para a descrição apresentada passasse a ser uma redução de Karp.

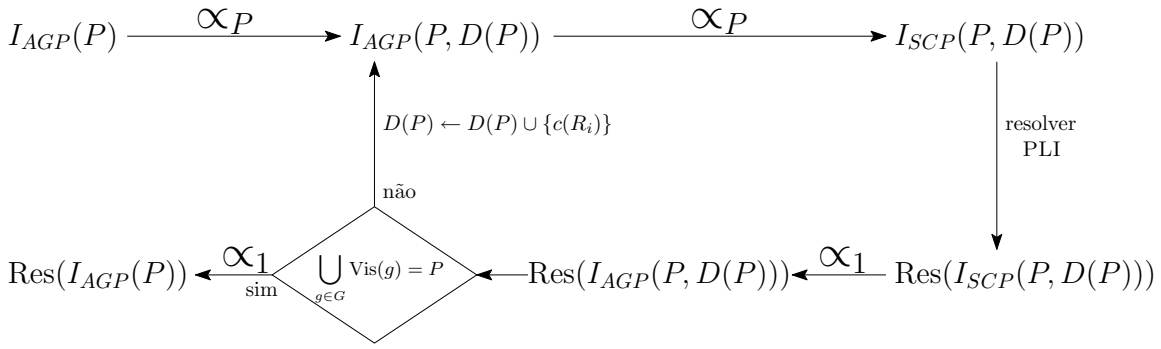


Figura 3.4: Esquematização da Redução: AGP $\preceq_P$ SCP.

3.4 Estratégias de Discretização

Inúmeras estratégias podem ser empregadas para a construção da discretização inicial $D(P)$ do polígono P. Para fins teóricos, os quais provam a exatidão e convergência do algoritmo proposto, é indiferente qual a discretização inicial utilizada. Entretanto, para fins práticos, notadamente no que diz respeito à eficiência do algoritmo, a escolha da estratégia de discretização é de extrema importância.

Lembremos que o número de iterações necessárias para que o algoritmo encontre uma solução ótima está estreitamente relacionado à existência de regiões não cobertas em cada iteração. Além disso, nessas iterações, uma instância de um problema NP-DIFÍCIL deve ser resolvida. Por isso, é importante que a estratégia de discretização inicial escolhida

gere uma instância "leve" do SCP e que, ao mesmo tempo, procure minimizar o número de iterações suficientes para que uma solução ótima seja encontrada.

Veja que há aqui um compromisso entre a velocidade em que se processa uma iteração do algoritmo e a qualidade da aproximação fornecida pela solução da discretização. São estes dois pontos que devem ser considerados ao se fazer a concepção de uma boa estratégia para a discretização inicial. Se por um lado, o uso de propriedades geométricas sofisticadas pode resultar em discretizações mais eficientes no que tange a qualidade das soluções do problema discreto, por outro lado, o custo de computação dessas propriedades pode superar o ganho da solução de um número menor de instâncias do problema NP-DIFÍCIL.

Esta seção mostra em detalhes as diferentes estratégias de discretização inicial do polígono P que foram investigadas nos artigos [16, 11, 14] publicados em decorrência da pesquisa realizada nesta dissertação. Tais artigos estão reproduzidos nos próximos capítulos e permitem que se tenha uma visão sobre como evoluiu o processo de concepção dessas estratégias, destacando os aspectos positivos e negativos de cada uma delas quando considerados os pontos expostos acima.

3.4.1 *Regular Grid* ("Grade Regular")

A estratégia de discretização denominada grade regular, foi utilizada no nosso primeiro trabalho [14] sob o termo *Regular Grid*. Baseada em discretizações propostas anteriormente na literatura [22, 23], essa estratégia tem por objetivo construir uma grade densa de pontos dentro do polígono. O objetivo é garantir que uma boa porção desse último seja vista por qualquer conjunto de vértices que guarda a discretização, o que, potencialmente, conduz a uma baixa quantidade de regiões não cobertas e a um pequeno número de iterações.

A grade é construída considerando-se todos os pontos de P que ocorrem na interseção de dois conjuntos de retas verticais e horizontais definidos como se segue. Inicialmente, define-se um passo de tamanho igual ao menor vão (intervalo consecutivo) das coordenadas x e y dos vértices do polígono P. Em seguida, atribui-se a um ponto p a menor ordenada e a menor abcissa do conjunto de vértices do polígono e, desenha-se a reta horizontal (vertical) passando pela ordenada (abcissa) de p e, sempre deslocando-se para cima (direita) de uma distância equivalente ao tamanho do passo, desenha-se uma nova reta paralela à anterior. A discretização inicial é completada com a inclusão de todos os vértices de P (veja detalhes Pg. 55). A Figura 3.5 mostra um exemplo da discretização inicial produzida por essa estratégia.

Durante a evolução dos trabalhos, percebeu-se que essa estratégia não era competitiva para instâncias de polígonos que não fossem as aleatórias ortogonais, pois, o número de pontos gerados para discretizar P crescia com quadrado do número de vértices, fazendo

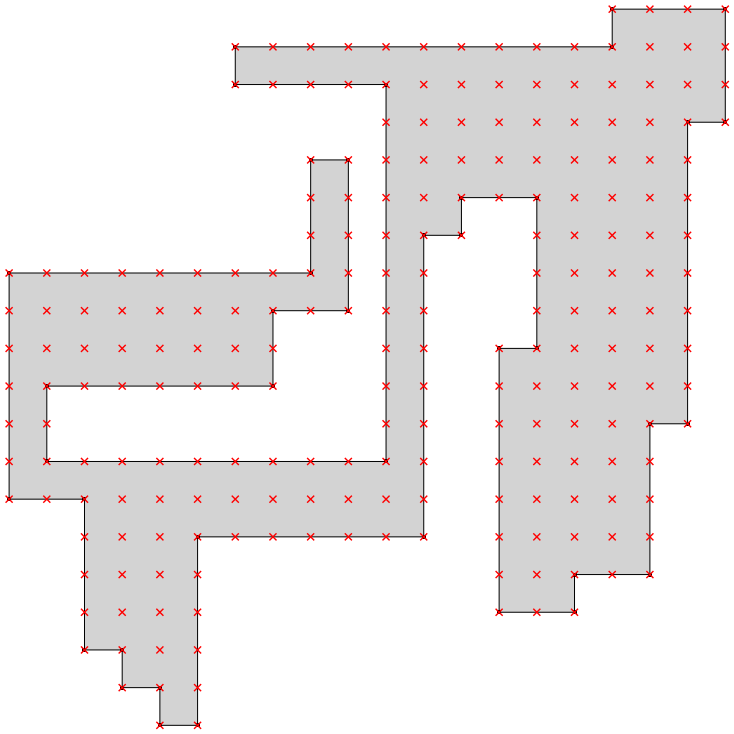


Figura 3.5: Exemplo da discretização inicial construída pela estratégia *Regular Grid*.

com que o modelo PLI do SCP ficasse demasiado grande, inviabilizando seu uso prático.

3.4.2 *Induced Grid* ("Grade Induzida")

Introduzida no trabalho [15] sob o termo *Induced Grid*, a estratégia de discretização grade induzida foi a primeira das estratégias pesquisadas a considerar uma propriedade geométrica do polígono, e assim tentar diminuir o número de iterações suficientes para atingir uma solução ótima do problema.

A grade induzida tem sua construção baseada no método desenvolvido por Culberson e Reckhow [18, 19]. Nele, o polígono P é particionado estendendo-se ambas as arestas adjacentes a cada vértice reflexo para dentro do polígono, até que seja atingido uma outra aresta da fronteira de P. Os segmentos de reta resultantes desta operação definem o arranjo que dá origem a grade induzida de P. Os pontos da discretização são aqueles que se encontram nas interseções desses segmentos. A Figura 3.6 mostra um exemplo da discretização inicial produzida por essa estratégia.

A estratégia se baseia na percepção de que os vértices reflexos são responsáveis por parte da dificuldade do problema já que a não convexidade de P na vizinhança desses vértices produz regiões que são visíveis por (potencialmente) poucos vértices do polígono.

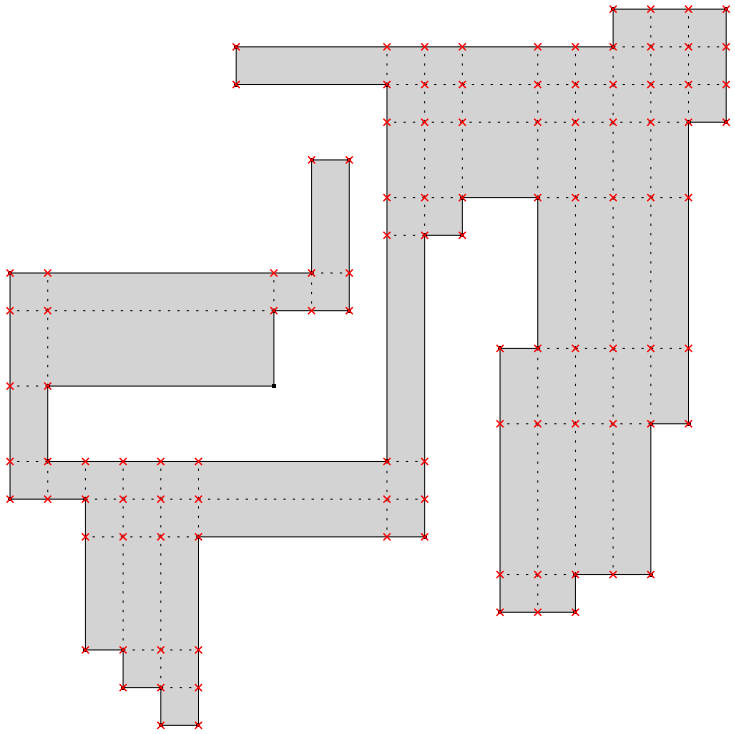


Figura 3.6: Exemplo da discretização inicial construa pela estratégia *Induced Grid*.

Tem-se então que um menor número de pontos são gerados na discretização em relação à estratégia anterior, ao mesmo tempo em que é almejada uma diminuição no número de iterações.

Embora essa estratégia gere um número menor ou igual de restrições ao modelo PLI que a estratégia da grade regular, ela se mostrou pouco competitiva durante a evolução dos trabalhos.

3.4.3 *Single Vertex* ("Vértice Único")

A estratégia de se utilizar um único vértice para a discretização inicial foi apresentada no último trabalho [11] sob a denominação *Single Vertex*.

À primeira vista essa estratégia parece bastante ingênua e com poucas chances de sucesso no que tange a eficiência. Isso se deve à pouca informação geométrica sobre o polígono que um único vértice carrega, e também à grande quantidade de regiões que as soluções iniciais tendem a deixar descobertas. Todavia, a idéia principal por trás dessa estratégia é poupar ao máximo o tempo despendido na resolução das formulações PLI mantendo-as o mais leve possível.

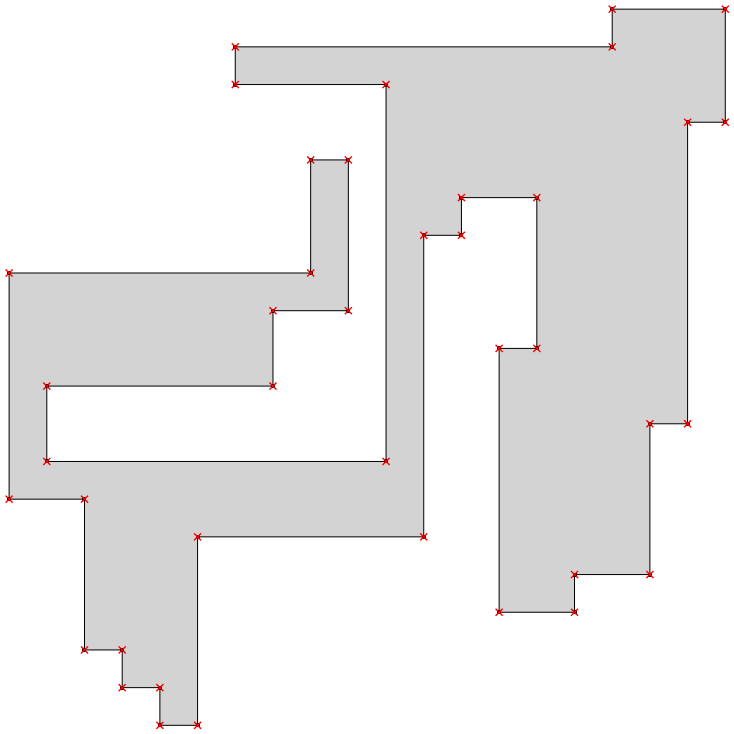


Figura 3.7: Exemplo da discretização inicial construída pela estratégia *All Vertices*.

3.4.4 *All Vertices* ("Todos os Vértices")

A estratégia de discretização denominada todos os vértices, foi inicialmente utilizada no trabalho [15] sob o termo *Just Vertices*. Após o estudo de outras estratégias que também se iniciavam com conjuntos de vértices de P, passou-se a usar, para esta, o termo *All Vertices*.

A Figura 3.7 mostra um exemplo de como esta estratégia gera uma discretização que, embora esparsa, possibilita a cobertura de muitas regiões que estão contidas em poucas regiões de visibilidade.

A razão de ter uma estratégia que contenha todos os vértices do polígono é justamente a de explorar o fato que os vértices capturam grande quantidade de informação geométrica sobre a forma de P a um custo desprezível de preprocessamento. Os experimentos mostram que as regiões não cobertas nesse caso são menores que aquelas obtidas pela estratégia anterior, o que faz com que as restrições adicionadas ao modelo PLI contribuam para que regiões de pouca visibilidade sejam mais facilmente cobertas, levando a que soluções melhores emerjam em poucas iterações.

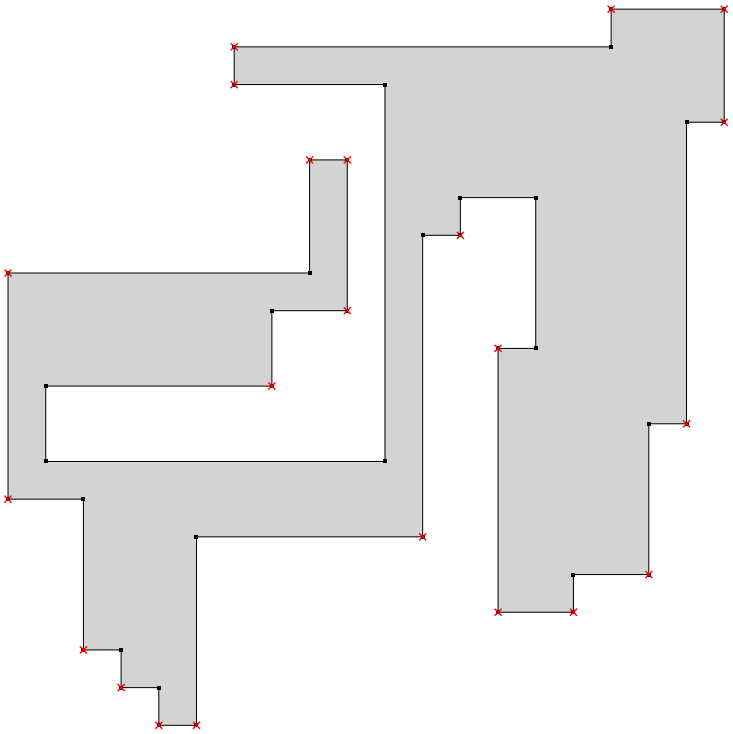


Figura 3.8: Exemplo da discretização inicial construída pela estratégia *Convex Vertices*.

3.4.5 *Convex Vertices* ("Vértices Convexos")

A estratégia de se utilizar o subconjunto dos vértices convexos de P para a discretização inicial foi apresentada nos últimos trabalhos [11, 12, 13] sob a denominação *Convex Vertices*.

Claramente, os vértices convexos são mais úteis para a discretização que os reflexos, visto que esses últimos são mais fáceis de serem vistos que os outros. Portanto, se algum subconjunto dos vértices contém alguma informação redundante e superflua sobre o polígono, é natural assumir que sejam os reflexos, o que suporta a escolha da estratégia *Convex Vertices*.

A Figura 3.8 mostra um exemplo da discretização inicial produzida por essa estratégia. Note que, a um custo baixo de preprocessamento, o tamanho da discretização foi significativamente reduzido se comparado com a estratégia anterior, enquanto se reteve ainda parte substancial das informações sobre a forma geométrica de P.

3.4.6 AVPs

Embora presente em todos os trabalhos apresentados para provar o conceito de convergência do algoritmo proposto, os AVPs só vieram a ser utilizados em uma estratégia

de discretização a partir de [15], sob a denominação *Complete Atomic Visibility Polygons*.

Note que a idéia guia para construção da discretização inicial das estratégias anteriores era ou torná-la densa e, com isso, capturar determinadas propriedades do polígono, ou torná-la esparsa e, com isso, resolver a instância do SCP de forma mais rápida ainda que por um número maior de vezes. Na estratégia AVP, a idéia é fazer uma única iteração para que somente uma instância de um problema NP-DIFÍCIL tenha que ser resolvida.

De acordo com o que foi exposto na subseção 3.2.3, quando um guarda vê um ponto qualquer dentro de um AVP, ele irá, necessariamente, ver o AVP inteiro. Posto isso, ao utilizar o arranjo de visibilidade dos vértices para construir uma discretização inicial que tenha um ponto dentro de cada AVP, escolhemos o seu centróide, garante-se que a solução do problema discreto será viável após a primeira iteração.

A Figura 3.9 mostra um exemplo da discretização gerada por essa estratégia. Repare que o arranjo de visibilidade está presente e que há exatamente um ponto da discretização para cada AVP.

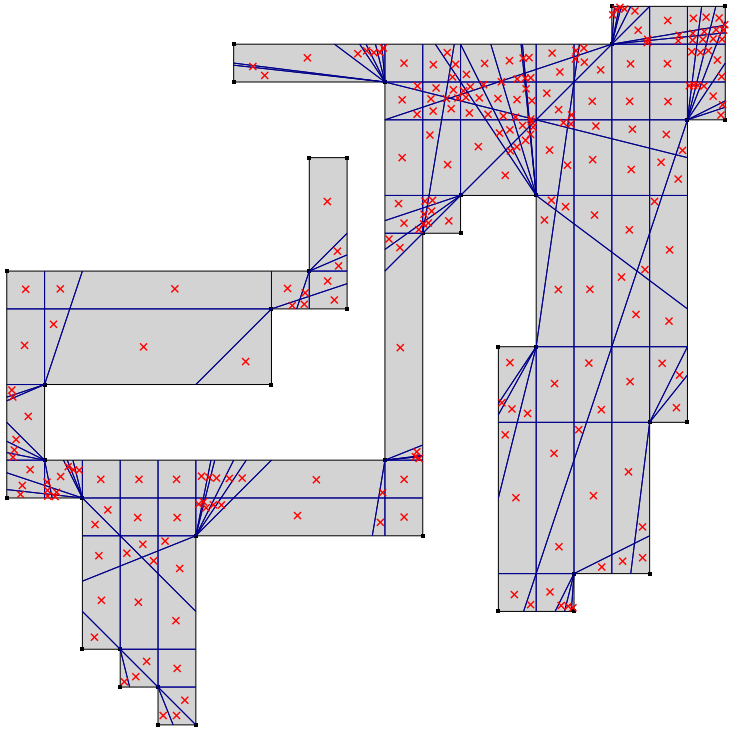


Figura 3.9: Exemplo da discretização inicial construída com todos os AVPs.

Entretanto, como pode haver $O(n^3)$ AVPs, o número de restrições do modelo PLI do SCP pode ser $O(n^3)$. Isso faz com que o uso prático dessa discretização seja viável apenas para galerias pouco complexas.

Felizmente, como mostrado a seguir, nem todos os AVPs são necessários para garantir

que o algoritmo faça somente uma iteração, o que torna essa estratégia dispensável.

3.4.7 *Shadow* AVPs ("AVPs de Sombra")

Como citado anteriormente, é possível reduzir significativamente o número de AVPs necessários na discretização acima e ainda garantir que o algoritmo encontre uma solução ótima para o problema contínuo após a primeira iteração. Essa estratégia consiste em colocar o centróide de todos os AVPs de sombra em $D(P)$.

Um AVP $\mathcal{S}$ é dito um AVP de sombra se existe um conjunto de vértices $W \subset V$ tal que $\mathcal{S}$ não seja visível por nenhum vértice em W e que toda aresta de $\mathcal{S}$ consiste de uma porção de uma aresta de P ou de uma região de visibilidade de algum vértice de W.

A estratégia AVPs de sombra consiste em utilizar o centróide de todos os AVPs de sombra para compor a discretização inicial (veja Figura 3.10).

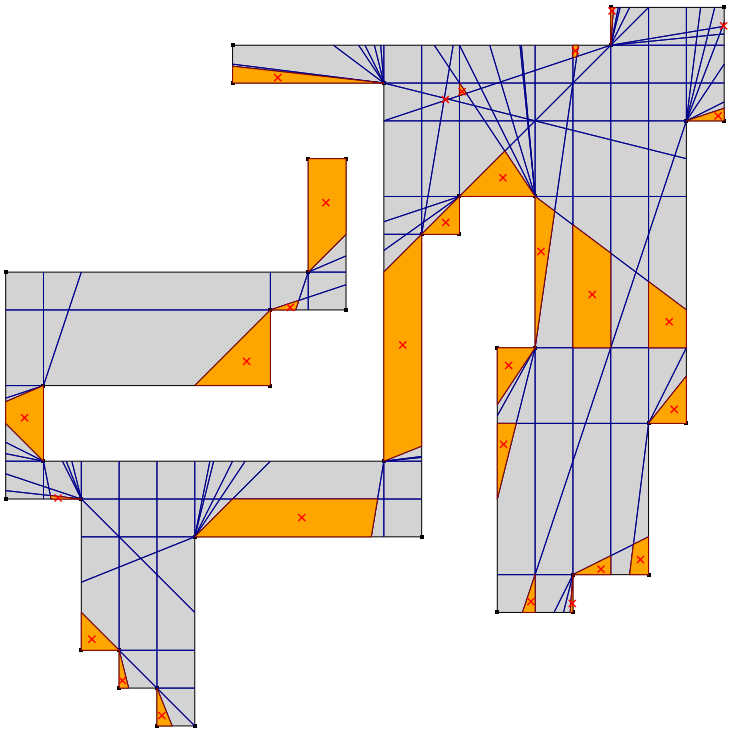


Figura 3.10: Arranjo de visibilidade, destacando os AVPs de sombra e os pontos da discretização inicial gerados pela estratégia *Shadow* AVPs.

Certamente é necessário mostrar que ao se utilizar o centróide de todos os AVPs de sombra como a discretização inicial $D(P)$, tem-se que a solução do algoritmo proposto será viável após a primeira iteração.

Teorema 3.4.1. *Seja $I(P, D(P))$ uma instância do Problema de Galeria de Arte discretizado para o polígono P onde $D(P)$ é o conjunto dos centróides de todos os AVPs de*

sombra de P. Tem-se então que, G é um CVG para P se e somente se G for um conjunto de vértices que guarda $D(P)$.

Prova. A parte da necessidade é trivial visto que $D(P) \subset P$. Portanto, focamos na prova da suficiência.

Suponha que $G \subset V$ seja um conjunto de vértices que guarda $D(P)$, mas não P. Então, existe uma região de P que não é coberta por nenhum dos vértices de G. Seja R a região conexa maximal não coberta por G. Repare que R é a união de AVPs disjuntos.

Para provar que ao menos um desses AVPs é um AVP de sombra, note que toda a região R não é vista por nenhum vértice em G cujas arestas das regiões de visibilidade que não façam parte de P geram R. Portanto, se R for um AVP, por definição ele é um AVP de sombra. Caso contrário, existe um vértice $v_i \in V$ que possui uma aresta e_{vi} de sua região de visibilidade que não é uma aresta de P mas que particiona R em duas regiões. Uma dessas regiões está do lado de e_{vi} que é visto por v_i, enquanto a outra não. Se atualizarmos R como sendo a sua parte não visível por v_i, através de um argumento indutivo, tem-se que ao particionar sucessivamente R da maneira apresentada, concluímos que ao menos um AVP de sombra está contido em R e, portanto, é não coberto.

Isso contradiz a hipótese de que G é um conjunto de vértices que guarda $D(P)$, já que esse é composto pelos centróides de todos os AVPs de sombra. □

Do Teorema 3.4.1, fica claro que ao utilizar a estratégia para construir a discretização inicial $D(P)$ tem-se que o algoritmo irá convergir em apenas uma iteração. Além disso, o tamanho da discretização diminui consideravelmente, se comparado com a estratégia anterior, tornando a utilização prática dos *Shadow* AVPs bastante mais atraente. Por outro lado, veja que o custo adicionado ao preprocessamento cresce significativamente devido à necessidade de computadas todas as faces do arranjo de visibilidade determinar quais delas são AVPs de sombra.

Capítulo 4

An Exact and Efficient Algorithm for the Orthogonal Art Gallery Problem

Prólogo

O artigo incluído neste capítulo foi escrito em co-autoria com os professores doutores Cid C. de Souza e Pedro J. de Rezende, ambos do Instituto de Computação da Universidade Estadual de Campinas. Apresentado em Belo Horizonte em outubro de 2007, durante o XX SIBGRAPI (*Brazilian Symposium on Computer Graphics and Image Processing*) e publicado pela *IEEE Computer Society* nos anais do mesmo simpósio, o artigo reflete o início da pesquisa apresentada nessa dissertação. Os objetivos almejados e alcançados naquele momento foram essencialmente de anunciar um algoritmo exato para o OAGP que superasse, em muito, os algoritmos existentes e detalhar uma técnica não muito difundida de trabalhar sobre o AGP mesclando ambas as óticas geométrica e combinatória do problema para a construção de um algoritmo robusto. O foco deste artigo foi o de apresentar detalhes do algoritmo, sua corretude, a prova de sua convergência e a viabilidade prática. Foi proposta também uma sugestão inicial para a discretização da galeria e, experimentalmente, verificou-se que o número de iterações necessárias para resolver à otimalidade as instâncias de teste é bem inferior ao limite teórico.

Abstract

In this paper, we propose an exact algorithm to solve the Orthogonal Art Gallery problem in which guards can only be placed on the vertices of the polygon P representing the gallery. Our approach is based on a discretization of P into a finite set of points in its interior. The algorithm repeatedly solves an instance of the Set Cover problem obtaining a minimum set Z of vertices of P that can view all points in the current discretization. Whenever P is completely visible from Z, the algorithm halts; otherwise, the discretization is refined and another iteration takes place. We establish that the algorithm always converges to an optimal solution by presenting a worst case analysis of the number of iterations that could be effected. Even though these could theoretically reach $O(n^4)$, our computational experiments reveal that, in practice, they are linear in n and, for $n \leq 200$, they actually remain less than three in almost all instances.

C1
(Pg.68)

Furthermore, the low number of points in the initial discretization, $O(n^2)$, compared to the possible $O(n^4)$ atomic visibility polygons, renders much shorter total execution times. Optimal solutions found for different classes of instances of polygons with up to 200 vertices are also described.

4.1 Introduction and Related Work

The classical *Art Gallery Problem* originally posed by Victor Klee in 1973 asked for determining the minimum number of guards sufficient to cover the interior of an n-wall art gallery [12]. Soon thereafter, Chvátal established what became known as *Chvátal's Art Gallery Theorem*: $\lfloor \frac{n}{3} \rfloor$ guards are occasionally necessary and always sufficient to cover a simple polygon with n vertices [3]. A simpler proof based on polygon triangulation and on the fact that this triangulation can be 3-colored was shown by Fisk in 1978 [10].

Many variations of the art gallery problem have been studied in the literature. Early on, Lee and Lin [17] proved that the vertex guards version is NP-hard by reduction from 3-SAT. Their result was extended to point guards by Aggarwal [1].

Since these early results, much research on related problems has been carried out by mathematicians and computer scientists. For broad surveys, the reader is referred to [18, 21, 25] where comprehensive analysis of many variations and results on this subject can be found.

In this paper, we study a variation of the classical art gallery problem, called *Orthogonal Art Gallery Problem*, which is an important subclass, due to most real life buildings and galleries being orthogonally shaped [25]. This problem deals specifically with the case where the guards can only be placed on the vertices of the polygon that defines the outer boundary of the gallery and whose edges are parallel to the x or y axis.

The earliest major result concerning this problem, due to Kahn *et al.* [15], states that $\lfloor \frac{n}{4} \rfloor$ guards are occasionally necessary and always sufficient to cover an orthogonal polygon with n vertices. Their proof is based on quadrilateralization and on a 4-coloring of the resulting graph.

More importantly, Schuchardt and Hecker proved that minimizing the number of guards in this variation is also NP-hard [20], settling a question that remained open for almost a decade [19].

Improvements on the efficiency of algorithms that place exactly $\lfloor \frac{n}{4} \rfloor$ guards, such as Edelsbrunner *et al.* [6] and Sack and Toussaint [19] who showed a linear time guard placement algorithm for monotone orthogonal polygons as well as an $O(n \log \log n)$ time algorithm for arbitrary orthogonal polygons, still do not treat the challenging problem of minimization.

In this line, as asserted by Urrutia [25], one approach that has not been sufficiently undertaken in the study of art gallery problems, is the one of finding algorithms that seek approximate solutions.

One of the first known results on this topic, due to Ghosh [11] in 1987, is an $O(n^5 \log n)$ time approximation algorithm, that finds a vertex guard set that is at most $O(\log n)$ times the minimum number of vertex guards needed, regardless of whether the polygon contains holes or not. Ghosh's work was later extended by Eidenbenz (see [7]) who designed approximation algorithms and heuristics for several variations of terrain guarding problems.

Recently, Erdem and Sclaroff in [8, 9] and Tomás *et al.* in [23, 24] modeled the problem as a discrete combinatorial problem and then solved the corresponding optimization problem. The results we present here follow this line.

Our Contribution. In this paper, we propose an algorithm to find a solution to the *Orthogonal Art Gallery Problem* by solving and refining discretizations of the boundary polygon. We show that the algorithm always produces a correct solution and we demonstrate, by a sizeable experimental analysis of its performance, that this algorithm derives its notable efficiency from the fact that the number of iterations required to achieve an optimal solution is very small, together with the fact that we deal with relatively few discretized points. We also present additional results regarding its implementation.

4.2 Preliminaries

In this paper, we address the *Orthogonal Art Gallery Problem* in which, given an orthogonal simple polygon that bounds an art gallery, the goal is to determine the minimum number and an optimal placement of (vertex) guards so as to keep the whole gallery under surveillance. In order to establish a uniform notation, whenever we speak of an

n-wall orthogonal art gallery we are referring to a planar region whose boundary consists of an orthogonal simple polygon (without holes), i.e., one whose n edges are parallel to the x or y axis. Given one such polygon P, we denote by V the set of vertices of P. A vertex $v \in V$ is called *reflex* if the internal angle at v is greater than $180\,^{\circ}$. Whenever no confusion arises, *a point in* P will mean a point either in the interior or on the boundary of P.

Given two points x and y in a simple polygon P, we say that y is visible from x if and only if the closed segment $\overline{xy}$ does not intersect the exterior of P. The set $V(v)$ of all points of P visible from a vertex $v \in V$ is called the *visibility region of* v. To determine the visibility region of a vertex v we employ the linear time algorithm proposed by Lee [16] and extended by Joe and Simpson [13, 14].

A set of points G is a *guard set* for P if for every point $p \in P$ there exists a point $g \in G$ such that p is visible from g. In other words, a guard set for P gives the positions of stationary guards who can oversee an entire art gallery of boundary P. Hence, the Orthogonal Art Gallery problem amounts to finding the smallest subset $G \subset V$ that is a guard set for P.

4.2.1 The discretized orthogonal art gallery problem

Approximate solutions to the Orthogonal Art Gallery problem can be obtained by modelingit as a discrete combinatorial problem, the *Minimum Set Cover problem*, as shown by Erdem and Sclaroff in [8, 9] and by Tomás *et al.* in [23, 24]. Both approaches discretize the polygon P in some way and then solve the corresponding optimization problem. The latter uses Constraint Programming to solve the problem, while the former applies Integer Programming.

In our formulation, we discretize polygon P into a set of points $D(P)$ and use a simplification model from [8, 9] as follows.

$$
\begin{aligned}
z \;=\; \min \quad & \sum_{j \in V} x_j \\
\text{s.t.} \quad & \sum_{j \in V} a_{ij} x_j \geq 1, \forall p_i \in D(P) \quad (1) \\
& x_j \in \{0,1\}, \forall j \in V \qquad (2)
\end{aligned}
$$

where

$$a_{ij} = \begin{cases} 1, \text{ if } p_i \in V(j) \\ 0, \text{ otherwise.} \end{cases}$$

$$x_j = \begin{cases} 1, \text{ if } j \text{ belongs to the solution} \\ 0, \text{ otherwise.} \end{cases}$$

The set $Z = \{j \in V \mid x_j = 1\}$ is called the solution set. Constraint (1) states that each point $p_i \in D(P)$ is visible from at least one selected guard position in the solution and the objective function minimizes the cardinality z of the solution set Z.

Thus, instead of solving the problem based on the entire polygon P, a discretization $D(P)$ is employed to generate a small number of constraints in the formulation. Clearly, it may happen that the solution of the discrete problem does not form a guard set for P. However, one can think of Z as an approximation to the original problem whose quality can be measured by the area of the regions not visible from the points in Z. Furthermore, the uncovered regions depend on the choice of the discretization. Among the various possible methods to discretize P, we focus on and extend the approach from [8] in which a regular grid is built prior to sampling the polygon. Refining the grid increases its resolution but also increases the number of constraints in (1). This constitutes an interesting tradeoff between speed and accuracy for the discrete approximation. The way we address this issue in our algorithm is further discussed below.

4.2.2 Initial discretization of P

We now describe how we build the first discretization of P. Consider the regular grid with resolution $\Delta_x \times \Delta_y$ starting at the lower left corner of the bounding box of P, where
$\Delta_x = min\{|v_x - u_x|,\ v_x \neq u_x\ :\ u, v \in V\}$ and
$\Delta_y = min\{|v_y - u_y|,\ v_y \neq u_y\ :\ u, v \in V\}$.

The intersection points of this grid that are interior to P form the initial discretization $D(P)$ of P. Below, we justify why this is a particularly good choice.

Consider the method developed by Culberson and Reckhow [4, 5] in which one partitions the interior of P by extending the two edges adjacent to each reflex vertex within the polygon until the first boundary edge of P is reached. These line segments induce a subdivision of P into rectangles, called *basic regions* of P.

In Figure 4.1, we illustrate these definitions by showing a polygon, within a 7×6 bounding box, whose basic regions are bounded by segments from the boundary of the polygon and by dashed lines. A regular grid of resolution 2×2 whose segments are represented by solid gray lines is superimposed.

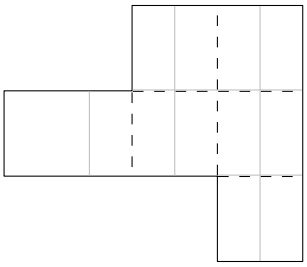


Figure 4.1: Regular grids and basic regions.

In the next proposition, we show that each basic region contains at least one point from the regular grid.

Proposition 4.2.1. *Let the regular grid $D(P)$ be the discretization of P with resolution $\Delta_x \times \Delta_y$ as defined above. Then, every basic region of P contains at least one point from $D(P)$.*

Proof. Let B be any basic region of P. From its construction, B is a rectangle with sides $\ell_x \geq \Delta_x$ and $\ell_y \geq \Delta_y$. Let LL be the lower-left and UR be the upper-right vertices of B. It follows that $UR_x \geq LL_x + \Delta_x$ and $UR_y \geq LL_y + \Delta_y$.

If LL is a vertex of P, let $p = LL$. Otherwise, let $p \in D(P)$ be the closest grid point to LL by the Manhattan (or L_1) distance dominated by LL, i.e., so that $p_x \leq LL_x$ and $p_y \leq LL_y$ (see Figure 4.2).

Now, from the definitions of Δ_x and Δ_y, the point $q = (p_x + \Delta_x, p_y + \Delta_y)$ is a grid point.

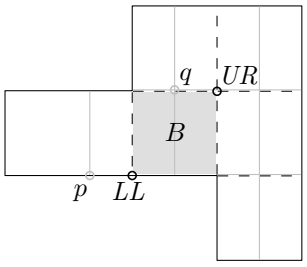


Figure 4.2: Illustration for the proof of Proposition 4.2.1.

Since $LL_x < q_x \leq LL_x + \Delta_x \leq UR_x$ and $LL_y < q_y \leq LL_y + \Delta_y \leq UR_y$, we have $q \in B$. □

Therefore, every basic region will have non empty intersection with the visibility polygon of at least one vertex guard from a solution to the discretized formulation, which makes the regular grid a good starting point to obtain an optimal solution.

Finally, we add to $D(P)$ all the vertices in V to form the complete discretization of the polygon P that we sought.

4.3 Algorithm

As mentioned earlier, a solution set Z to the discretized formulation in Section 4.2 may not always constitute a guard set for P since there might be regions inside P that are not visible from any guard in Z.

Definition 4.3.1. *Let $I(P, D(P))$ be an instance of the discretized Orthogonal Art Gallery problem with polygon P as the gallery boundary and $D(P)$ a discretization of P. A solution Z of this instance is called* viable *if Z is a guard set for P, i.e., $\bigcup_{g \in Z} V(g) = P$.*

Our algorithm overcomes the aforementioned drawback of the solution to the original discretized formulation and always produces a viable solution by refining the discretization of P whenever it detects that the present solution is not viable. The following theorem establishes that a solution thus obtained is also optimal.

Theorem 4.3.1. *Let Z be a solution of an instance $I(P, D(P))$ of the discretized Orthogonal Art Gallery problem. If Z is viable then Z is optimal.*

Proof. Due to the fact that Z is a solution of the minimization problem $I(P, D(P))$, Z is optimal as a vertex guard cover for the set $D(P)$ of points which discretize the polygon P, i.e., $z = |Z|$ is minimal among the cardinalities of all vertex guard covers of $D(P)$.

C2 (Pg.68)

Now, let Z^* be an optimal vertex guard set for P and let $z^* = |Z^*|$. Since Z^* is also a vertex guard cover for $D(P)$, we must have $z^* \geq z$. On the other hand, since Z^* is viable, it follows that $z \geq z^*$. □

Theorem 4.3.1 states that when the algorithm finds a solution for the discretized formulation which is viable, that solution is also an optimal vertex guard cover for P, i.e., it is a guard set for P.

The algorithm is divided into two phases: a Preprocessing Phase, where the initial discretization described in Section 4.2 is constructed and the Integer Programming problem is set up, and a Solution Phase in which the discretized problem is successively solved and refined until a viable (and optimal) solution is found.

4.3.1 Preprocessing Phase

In order to assemble the formulation outlined in Section 4.2, we start by building a regular grid of resolution $\Delta_x \times \Delta_y$ restricted to the interior of the polygon P, to which we add the vertices of P, as described in Section 4.2.2.

Once this discretization is built, we compute which grid points are located inside the visibility region of each vertex in V, and, then, include these new restrictions in the formulation.

The main steps of the preprocessing phase are summarized in Algorithm 4.3.1. If the regular $\Delta_x \times \Delta_y$ grid $D(BB)$ has m points, the generation of the initial discretization $D(P)$, in step 3, requires $O(mn)$ time when identifying the grid points that lie inside P. The total complexity of step 6 is $O(n^2)$ [13] and, assuming that $m \in \Omega(n)$, the full complexity of step 8 is $O(nm \log n)$ since point location of each of the $O(m)$ points of $D(P)$ in a star-shaped visibility n-polygon can be accomplished in $O(\log n)$ time. Hence, the overall complexity of the preprocessing phase is dominated by that of step 8.

Algorithm 4.3.1 Preprocessing Phase

1: $BB \leftarrow$ bounding box of P;
2: $D(BB) \leftarrow \Delta_x \times \Delta_y$ regular grid generated as described in Section 4.2.2;
3: $D(P) \leftarrow D(BB) \cap P$;
4: $D(P) \leftarrow D(P) \cup V$;
5: **for** each $j \in V$ **do**
6: Compute $V(j)$;
7: **for** each grid point $p_i \in D(P)$ **do**
8: $a_{ij} \leftarrow$ **Boolean**$(p_i \in V(j))$;
9: **end for**
10: **end for**

The result of the preprocessing phase is an Integer Programming (IP) formulation for the Set Cover Problem [26] which, once solved, generates a solution Z that, while not necessarily constituting a guard set for P, will always cover all the grid points in $D(P)$.

4.3.2 Solution Phase

In the second phase of the algorithm, starting from the IP formulation generated in the preprocessing step, we solve the discretized instance followed by a refinement of the grid iteratively until the solution becomes viable. This refinement is attained by generating one more point in the discretization for each uncovered region (its centroid) and by adding the corresponding constraints to the current Set Cover model. These additional points enhance the regular grid and lead to a solution closer to a viable one. Algorithm 4.3.2 outlines the steps executed in the solution phase.

It remains to be argued that Algorithm 4.3.2 converges, as it will then follow from Theorem 4.3.1 that the algorithm is exact and the solution given is indeed a guard set for P. In order to determine the worst case for the number of iterations done by the algorithm, we proceed as follows.

Algorithm 4.3.2 Solution Phase

1: **repeat**
2: $Z \leftarrow$ solution of $I(P, D(P))$;
3: **for** each uncovered region R **do**
4: $c \leftarrow$ centroid of R;
5: $D(P) \leftarrow D(P) \cup \{c\}$;
6: Add a new row, r, to the set of constraints (1)
corresponding to the new uncovered point c:
$a_{rj}x_j \geq 1$ where, $\forall j \in V$,
$a_{rj} \leftarrow$ **Boolean**$(c \in V(j))$;
7: **end for**
8: **until** Z is viable


C3
(Pg.68)

Consider the set of all visibility regions of the vertices in V, whose union, obviously, covers P. The edges of these visibility regions induce a subdivision of P which is comprised of what is called *atomic visibility polygons*, or AVPs [11] (see Figure 4.3). Note that in step 3: any uncovered region (witness to the fact that Z does not cover the entire polygon) is necessarily a simple polygon formed by the union of neighboring AVPs. Furthermore, it follows from the construction of the AVPs that if the centroid of (or, for that matter, any point within) an atomic visibility polygon $\mathcal{V}$ is visible from a vertex guard, the entire area of $\mathcal{V}$ must also be.

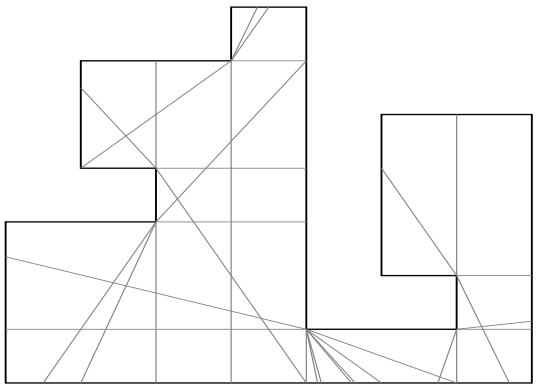


Figure 4.3: Atomic Visibility Polygons.

As the visibility region of any vertex can have at most $O(n)$ edges, the induced subdivision is generated from an arrangement of $O(n^2)$ lines and has a total complexity of no more than $O(n^4)$ faces (or AVPs).

Therefore, an upper bound on the maximum number of iterations effected by the algorithm is $O(n^4)$ and this establishes the convergence. Of course, in the worst case,

each iteration can take exponential time as step 2: amounts to solving an instance of the Set Cover Problem.

Note that if $D(P)$ were built by taking one interior point from each AVP, the algorithm would halt right after the first iteration. In this case, the total complexity would be $\mathrm{EXP}(n^4)$. On the other hand, by starting with $D(P)$ as described in Section 4.2.2, the complexity of the algorithm is bound by $\mathrm{EXP}(n^2)$, provided that the required number of iterations remains within $O(1)$. In Section 4.5, we show that this assumption is true in practice.

4.4 Experimental Setup

This section presents the experimental setup for performance evaluation of our method. For the experiments conducted, we implemented the algorithms described in Section 4.3 along with a visibility algorithm from [13]. The implementation was done in `C++` on top of the CGAL 3.2.1, and used the Integer Linear Programming solver Xpress-Optimizer `v17.01.02`. The algorithm was tested on a PC featuring a Pentium IV at 2.66 GHz and 1 GB of memory.

Three sessions of tests were carried out, one for each of the three classes of polygons discussed in Section 4.4.1. Each session consisted of solving the Orthogonal Art Gallery problem for multiple instances. In each test, we computed several measures: preprocessing time spent by Algorithm 4.3.1, execution time and the number of iterations performed by Algorithm 4.3.2, and the number of guards in the optimal solution. Lastly, we compiled this information to show evidence of the efficiency of our algorithm, as is presented in Section 4.5.

4.4.1 Instances

We conducted the experiments on a large number of instances which consisted of n-vertex orthogonal polygons placed on an $\frac{n}{2} \times \frac{n}{2}$ unit square grid. These polygons were generated devoid of collinear edges, in accordance to the method described in [22].

We generated a total of 10282 sample polygons, each one having between 8 and 200 vertices, classifiable in three classes introduced in [24]: *random, small area* and *large area* polygons. See samples in Figure 4.4. Large area and small area polygons are the extreme scenarios for the IP approach in terms of the number of constraints in the model (see [24]). On one hand, large area polygons give rise to a very dense initial grid, while small area polygons lead to very few grid points. Recalling that orthogonal polygons have an even number of vertices, the data set used in our experiments was obtained thus: (a) 10088 Random polygons: for each k between 8 and 200, we generated k distinct polygons, with

C4 (Pg.68)

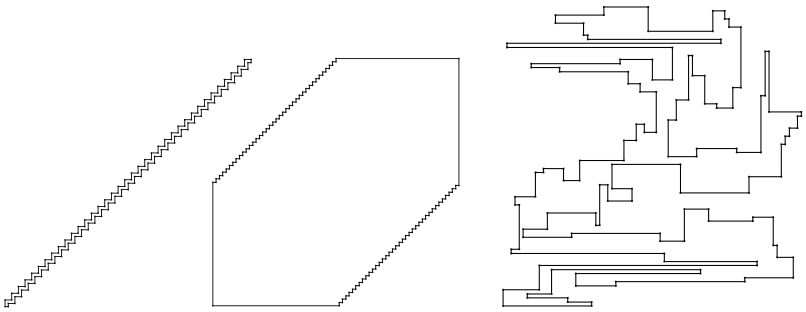


Figure 4.4: Sample polygons with 150 vertices: Small Area, Large Area and Random polygons.

k vertices in each, by applying the algorithm for random orthogonal polygon generation described in [22]; (b) 97 Small and 97 Large Area polygons: one for each number of vertices between 8 and 200.

4.5 Results

We now describe the experimental results that attest to the efficiency of the algorithm presented in Section 4.3. The testing was performed using the instances described in the previous section. Charts are employed to present plots of the data that summarize the statistical analysis.

Although we established that in the worst case the number of iterations of Algorithm 4.3.2 is bound by $O(n^4)$, in Figure 4.5 we show a histogram demonstrating that the practical results are quite impressive for polygons with 200 vertices or less. Indeed, the Boxplot [2] below reveals that in our experiments the median was only *one* iteration and that the third quartile is *two*. In fact, the algorithm finds an optimal solution for more than 99% of the instances within three iterations. This means that the instances of up to 200 vertices that need four or more iterations are outliers.

Figure 4.6 shows another interesting result about the number of iterations required to achieve optimal solution to the test instances. Notwithstanding the claim by Tomás *et al.* in [24] that large area and small area polygons constitute extremal cases, our algorithm vanquishes these, at least in regard to the number of iterations to achieve optimal solutions — which turns out to be *one*. A further analysis of Figure 4.6 also indicates that the growth of the average number of iterations to solve instances of random polygons is linear and slow growing on n as its linear regression $0.003 \cdot n + 1.1$ asserts.

In Figure 4.7 we plot the average size of the guard sets as a function of the number of vertices, n. Here, we have evidence that, as one would expect, the result by

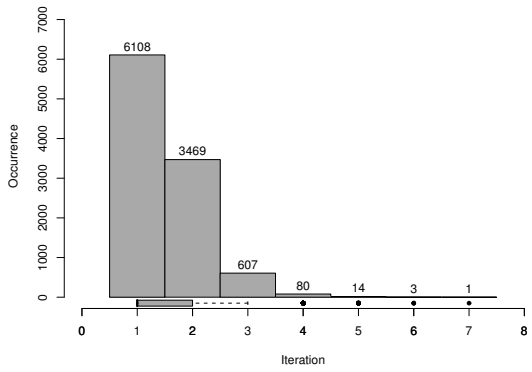


Figure 4.5: Histogram with a Boxplot showing how often each number of iterations occur.

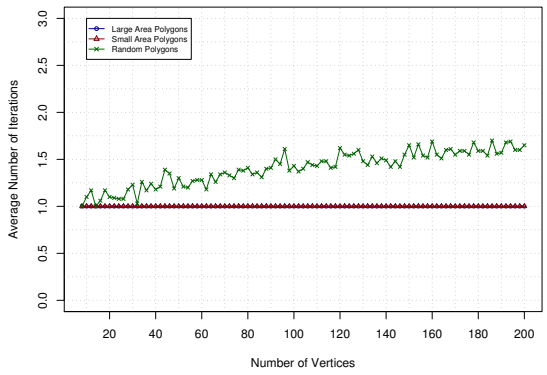


Figure 4.6: Average number of iterations to completely solve instances of different sizes.

Kahn *et al.* [15], which states that $\lfloor \frac{n}{4} \rfloor$ guards are occasionally necessary and always sufficient to cover an orthogonal polygon of n vertices, gives in practice a fairly distant upper bound on the average for these polygons. Figure 4.7 also shows that large area polygons need no more than *two* guards in their optimal solutions.

Timewise, the preprocessing phase, Algorithm 4.3.1, presents the behavior indicated in Figure 4.8, where it becomes evident that the large area polygons produce a much denser regular grid than the other two types of polygons. The higher growth in this case comes from the fact that, as the number of vertices of P increases, the combinatorial complexity of the regular grid grows even faster since the area of the polygons available for intersections in step 3 of Algorithm 4.3.1 is quadratic in n. Thus, the size of the initial discretization $D(P)$ is $\Theta(n^2)$ and, according to the complexity analysis done in Section 4.3,

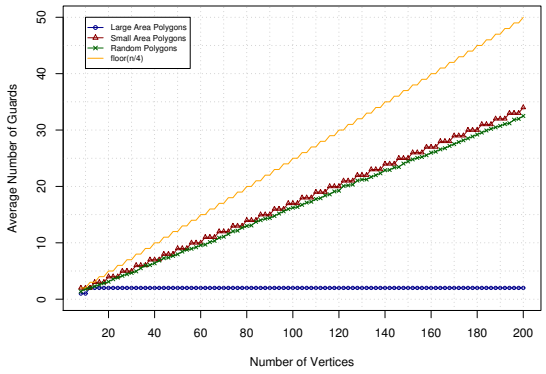


Figure 4.7: Average size of guard sets.

the expected running time of Algorithm 4.3.1 for these polygons is $O(n^3 \log n)$. The plot for large area polygons in Figure 4.8 is consistent with this result.

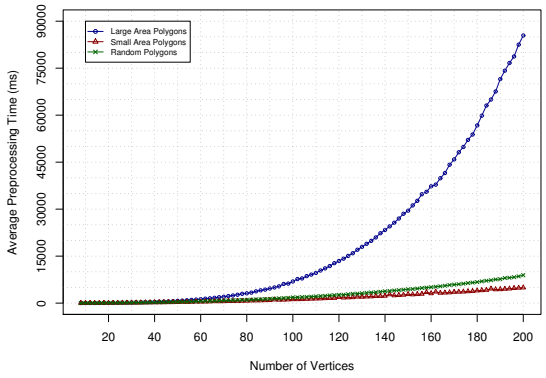


Figure 4.8: Average preprocessing time.

Lastly, Figure 4.9 shows the amount of time spent in the solution phase (Algorithm 4.3.2). For random polygons, the time taken grows almost linearly on the number of vertices because the average number of iterations needed to optimally solve those instances is also a linear function of n (refer back to Figure 4.6). One also notices that instances of small area polygons take only a negligible amount of time. Regarding large area polygons, the performance is hindered as a consequence of the higher density of the regular grid. A closer look at the results from the experiment with large area polygons revealed some interesting facts. Firstly, the solution phase described in Algorithm 4.3.2

always halted after *one* iteration. Secondly, Xpress, the IP solver used in step 2 of the algorithm, solves only one linear programming relaxation. This is because, using its internal heuristics, Xpress finds a feasible solution for the Set Cover instance which is proved to be optimal by the dual bound produced by the relaxation. As a result, when applied to large area polygons in our dataset, the solution phase of our algorithm only involved the solution of a linear programming problem with $O(n^2)$ constraints and $O(n)$ variables as well as the computation of the uncovered regions (step 3 of Algorithm 4.3.2). This explains why the graph of the running time of our algorithm for large area polygons has a polynomial shape. In fact, after a thorough investigation using regression analysis, we found that the polynomial that best fits the running time plot of our algorithm is given by $1.994\times10^{-7}n^5 - 7.920\times10^{-5}n^4 + 0.01238n^3 - 0.6864n^2 + 16.26n - 110.9$. The graph of this polynomial is displayed in Figure 4.9 and it should be remarked that even though its coefficients were obtained by considering only the observations for n ranging from 8 to 140, the resulting curve extrapolates well for the remaining of the observed range.

C5 (Pg.69)

Despite the high order of the polynomial, considering that the Art Gallery problem for Orthogonal polygons is NP-hard, it is remarkable that, on the average, the most difficult instances with nearly 200 vertices that we tested could still be solved to optimality within 97 seconds, with only 10% of that time taken by the solution phase.

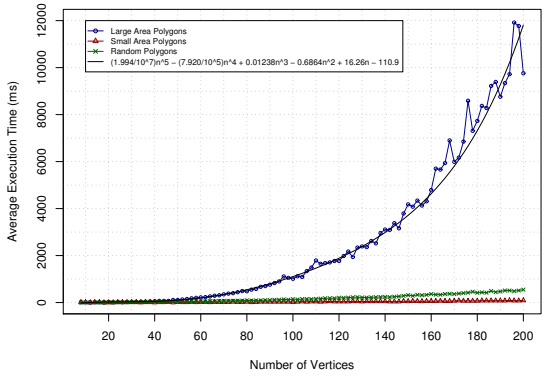


Figure 4.9: Average execution time.

4.6 Conclusion

We presented an exact and efficient algorithm for the *Orthogonal Art Gallery problem* when guards are restricted to the vertices of the gallery-bounding polygon. We also proved the theoretical results necessary to guarantee the convergence and the exactness

of the method. Even though we presented the algorithm to address polygonal galleries, the same method can be directly extended to terrains as discussed in [7].

Our computational tests were performed on a large number of polygons from three different classes and the results showed that our algorithm is very efficient in practice: more than 99% of the instances with up to 200 vertices were solved in no more than *three* iterations. In fact, we also obtained computational evidence to support the claim that the average number of iterations required by the algorithm to reach the optimal solution in any of those classes of orthogonal polygons is a slow linear function of the number of vertices. Furthermore, we were able to solve with a single iteration both classes of orthogonal polygons considered to possess extremal behavior in [24].

The tests also showed that the time spent by our algorithm in order to optimally solve the instances of random polygons is an almost linear function of the number of vertices; on the other hand, in the case of small area polygons, the algorithm uses no more than a negligible amount of time. Lastly, for large area polygons, the growth of the execution time as a function of the number of vertices turned out to be no more than a fifth degree polynomial.

Extensions to the present work will include the study of geometric properties that could lead to a reduction on the number of grid points and, hence, to a cutback of the preprocessing time and, as a consequence, to an improvement on the overall efficiency. Through further investigations we also intend to extend the method presented here to the case where guards can be placed on the interior of the edges of the polygons. One last obvious generalization is to obtain similar algorithms for non orthogonal polygons.

References

[1] A. Aggarwal, S. K. Ghosh, and R. K. Shyamasundar. Computational complexity of restricted polygon decompositions. In G. T. Toussaint, editor, *Computational Morphology*, pages 1–11. North-Holland, Amsterdam, Netherlands, 1988.

[2] J. M. Chambers, W. S. Cleveland, B. Kleiner, and P. A. Tukey, editors. *Graphical methods for data analysis*. The Wadsworth statistics/probability series. Wadsworth International, Duxbury Press, Boston, 1983.

[3] V. Chvátal. A combinatorial theorem in plane geometry. In *Journal of Combinatorial Theory Series B*, volume 18, pages 39–41, 1975.

[4] J. Culberson and R. Reckhow. Dent diagrams: A unified approach to polygon covering problems. Technical Report TR 87-14, Dept. Comput. Sci., Univ. Alberta, Edmonton, Alberta, Canada, 1987.

[5] J. Culberson and R. A. Reckhow. Covering a simple orthogonal polygon with a minimum number of orthogonally convex polygons. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 268–277, 1987.

[6] H. Edelsbrunner, J. O'Rourke, and E. Welzl. Stationing guards in rectilinear art galleries. *Comput. Vision Graph. Image Process.*, 27:167–176, 1984.

[7] S. Eidenbenz. Approximation algorithms for terrain guarding. *Inf. Process. Lett.*, 82(2):99–105, 2002.

[8] U. M. Erdem and S. Sclaroff. Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints. In *Proc. International Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras*, pages 30–41, 2004.

[9] U. M. Erdem and S. Sclaroff. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Comput. Vis. Image Underst.*, 103(3):156–169, 2006.

[10] S. Fisk. A short proof of Chvátal's watchman theorem. In *Journal of Combinatorial Theory Series B*, volume 24, page 374, 1978.

[11] S. K. Ghosh. Approximation algorithms for art gallery problems. In *Proc. Canadian Inform. Process. Soc. Congress*, 1987.

[12] R. Honsberger. *Mathematical Gems II.* Number 2 in The Dolciani Mathematical Expositions. Mathematical Association of America, 1976.

[13] B. Joe and R. B. Simpson. Visibility of a simple polygon from a point. Report CS-85-38, Dept. Math. Comput. Sci., Drexel Univ., Philadelphia, PA, 1985.

[14] B. Joe and R. B. Simpson. Correction to Lee's visibility polygon algorithm. *BIT*, 27:458–473, 1987.

[15] J. Kahn, M. M. Klawe, and D. Kleitman. Traditional galleries require fewer watchmen. *SIAM J. Algebraic Discrete Methods*, 4:194–206, 1983.

[16] D. T. Lee. Visibility of a simple polygon. *Comput. Vision, Graphics, and Image Process*, 22:207–221, 1983.

[17] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Trans. Inf. Theor.*, 32(2):276–282, 1986.

[18] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[19] J.-R. Sack and G. T. Toussaint. Guard placement in rectilinear polygons. In G. T. Toussaint, editor, *Computational Morphology*, pages 153–175. North-Holland, Amsterdam, Netherlands, 1988.

[20] D. Schuchardt and H.-D. Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Mathematical Logic Quarterly*, 41:261–267, 1995.

[21] T. C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[22] A. P. Tomás and A. L. Bajuelos. Generating random orthogonal polygons. In *Current Topics in Artificial Intelligence*, volume 3040 of *Lecture Notes in Computer Science*, pages 364–373. Springer Berlin / Heidelberg, 2004.

[23] A. P. Tomás, A. L. Bajuelos, and F. Marques. Approximation algorithms to minimum vertex cover problems on polygons and terrains. In *Proceedings of the International Conference on Computational Science (ICCS 2003)*, volume 2657 of *Lecture Notes in Computer Science*, pages 869–878. Springer Berlin / Heidelberg, 2003.

[24] A. P. Tomás, A. L. Bajuelos, and F. Marques. On visibility problems in the plane - solving minimum vertex guard problems by successive approximations. In *Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics (AI & MATH 2006)*, 2006. to appear.

[25] J. Urrutia. Art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. North-Holland, 2000.

[26] L. A. Wolsey. *Integer Programming*. Wiley-Interscience, 1998.

Comentários

C1: Limite Teórico do Algoritmo

O artigo apresenta o limite teórico do número de iterações do algoritmo como sendo $O(n^4)$. Estabelece-se esse limite considerando que gera-se, em cada iteração, exatamente um ponto dentro de cada AVP. Por definição, os AVPs são as regiões geradas pelo arranjo de visibilidade dos vértices da galeria e, como exitem n vértices e cada região de visibilidade possui $O(n)$ arestas, esse arranjo é gerado por $O(n^2)$ linhas, o que produz uma estimativa máxima de $O(n^4)$ faces ou AVPs.

Entretanto, um resultado de Bose *et al.* [9] prova que esse limitante não está justo e que um limitante mais próximo é $O(n^3)$. Essa pequisa faz com que o número máximo de AVPs e, portanto, o limite teórico do algoritmo apresentado, seja $O(n^3)$ ao invés de $O(n^4)$. É importante notar que, experimentalmente, o número de iterações necessárias ainda permanece ordens de grandeza inferior ao limite teórico.

C2: Prova do Teorema 4.3.1

O resultado obtido com esse teorema é de grande importância para o trabalho, pois permite que se possa resolver e encontrar uma solução ótima para o problema contínuo sem necessitar dos infinitos pontos do polígono como testemunhas, na formulação PLI. Para uma demostração mais detalhada desse teorema, ver Capítulo 3, página 36.

Há uma errata a ser feita na demostração desse teorema. Onde está escrito “On the other hand, since Z^* is viable”, leia-se “On the other hand, since Z is viable”.

C3: Errata do Algoritmo 4.3.2

Substitua a linha 6 do algoritmo por

Add a new row, r, to the set of constraints (1)
corresponding to the new uncovered point c:
$\sum_{j \in V} a_{rj} x_j \geq 1$ where $a_{rj} \leftarrow \textbf{Boolean}(c \in V(j)), \forall j \in V$;

C4: Classes de Polígonos

As instâncias utilizadas para a experimentação dos artigos estão descritas em detalhes no Capítulo 2, Seção 2.4, onde encontram-se as características de cada tipo de polígono e a regra de geração de cada um. Todas as instâncias que foram utilizadas nos testes do algoritmo e os resultados obtidos podem também ser baixados do portal *online* `www.ic.unicamp.br/~cid/Problem-instances/Art-Gallery`

C5: *Polynomial Time Fit*

Embora seja resolvido no processamento um problema NP-DIFÍCIL, a curva gerada pelos experimentos se assemelhou em muito àquela do preprocessamento, onde a complexidade de tempo encontrada nos resultados apresentados é polinomial e determinada por $O(n^3 \log n)$. Este fato acarreta surpresa sobre problemas NP-DIFÍCEIS, que, para determinadas instâncias práticas, podem ser resolvidos em tempo aceitável. Em particular, a conveniência de se utilizar um resolvedor PLI para o AGP deve-se ao fato de que, atualmente, os resolvedores estão adaptados para resolver instâncias do SCP, em especial as "bem comportadas", em baixíssimo tempo. A experimentação mostra que na prática, o gargalo do algoritmo é o preprocessamento polinomial, ao invés da expectativa mais lógica que seria o processamento exponencial. Um exemplo adicional a essa discussão é o fato de que o Simplex para PLI é exponencial, enquanto o Método dos Elipsóides é polinomial, mas, na prática, para muitas instâncias, as implementações com Simplex são bem mais eficientes do que aquelas que usam o Método dos Elipsóides.

Posta esta discussão, foi feito o *fit* polinomial, utilizando o pacote estatístico apresentado na Seção B.2.5 e obteve-se a curva do processamento, apenas dos valores no intervalo de $[8 - 140]$ vértices. Após, foi feita uma extrapolação dessa curva, e verificou-se que os valores obtidos para o restante do intervalo faziam um *fit* perfeito com a curva plotada.

Capítulo 5

Experimental Evaluation of an Exact Algorithm for the Orthogonal Art Gallery Problem

Prólogo

Neste capítulo, encontra-se o artigo apresentado em maio/junho de 2008 em Massachusetts – EUA, no 7^{th} WEA (*International Workshop on Experimental Algorithms*) e publicado pela *Springer* no volume 5038 do *Lecture Notes in Computer Science*. Este artigo foi escrito em co-autoria com os professores doutores Cid C. de Souza e Pedro J. de Rezende, ambos do Instituto de Computação da Universidade Estadual de Campinas. Ele reflete uma fase intermediária da pesquisa, onde ainda trabalhava-se com polígonos ortogonais, mas já eram estudadas alternativas para a discretização inicial da galeria. Algumas dessas alternativas são evoluções diretas de estruturas apresentadas no trabalho anterior, como a Grade Induzida e os AVPs. Dois grandes objetivos foram almejados e alcançados com este trabalho. O primeiro foi o de introduzir alternativas para a discretização inicial da galeria e o segundo foi o de apresentar novas classes de polígonos ortogonais, que se mostraram, experimentalmente, mais complexos e difíceis de resolver do que os do trabalho anterior. Um foco do artigo foi a introdução das alternativas de discretização, em especial da estratégia *All Vertices*. Esta tenta ao mesmo tempo ser simples, de baixa cardinalidade e conter relevantes propriedades geométricas do polígono. Outra estratégia é a dos *Shadow AVPs* que possui propriedade de reduzir em ordens de grandeza a quantidade de regiões onde deve existir um ponto da discretização para que o problema seja resolvido em apenas uma iteração. Outro foco do artigo foi o aumento no tamanho das instâncias que foram resolvidas à otimalidade pelo algoritmo proposto. Elas são cinco vezes maior do que as reportadas na literatura até então.

Abstract

We consider the Orthogonal Art Gallery problem (OAGP) whose goal is to minimize the number of vertex guards required to watch an art gallery whose boundary is an n-vertex orthogonal polygon P. Here, we explore an exact algorithm for OAGP, which we proposed in [1], that iteratively computes optimal solutions to Set Cover problems (SCPs) corresponding to discretizations of P. While it is known [1] that this procedure converges to an exact solution of the original continuous problem, the number of iterations executed is highly dependent on the way we discretize P. Although the best theoretical bound for convergence is $\Theta(n^3)$ iterations, we show that, in practice, it is achieved after only a few of them, even for random polygons of hundreds of vertices. As each iteration involves the solution of an SCP, the strategy for discretizing P is of paramount importance. In this paper, we carry out an extensive empirical investigation with five alternative discretization strategies to implement the algorithm. A broad range of polygon classes is tested. As a result, we are able to significantly improve the performance of the algorithm, while maintaining low execution times, to the point that we achieve a fivefold increase in polygon size, compared to the literature.

C1 (Pg.87)

5.1 Introduction

The classical *Art Gallery Problem* originally posed by Victor Klee in 1973 consists in determining the minimum number of guards sufficient to cover the interior of an n-wall art gallery [2]. Chvátal showed, in what became known as *Chvátal's Art Gallery Theorem*, that $\lfloor n/3 \rfloor$ guards are occasionally necessary and always sufficient to cover a simple polygon with n vertices [3].

Many variants of the art gallery problem have been studied in the literature. In this paper, we study the variation of the classical art gallery problem that deals specifically with orthogonal polygons (edges parallel to the x or y axis) where guards can only be placed on vertices that define the outer boundary of the gallery. This is called the *Orthogonal Art Gallery Problem* (OAGP) and is an important subclass, due to most real life buildings and galleries being orthogonally shaped [4].

The earliest major result concerning this problem, due to Kahn *et al.* [5], states that $\lfloor \frac{n}{4} \rfloor$ guards are occasionally necessary and always sufficient to cover an orthogonal polygon with n vertices. Later, Schuchardt and Hecker proved that minimizing the number of guards in this variation is also NP-hard [6], settling a question that remained open for almost a decade [7].

Several placement algorithms have been proposed in the past, such as Edelsbrunner *et al.* [8] and Sack and Toussaint [7], which deal with the problem of efficiently placing

exactly $\lfloor n/4 \rfloor$ guards covering a given orthogonal gallery.

On the other hand, in a recently revised manuscript, based on [9], Ghosh presents an $O(n^4)$ time approximation algorithm for simple polygons yielding solutions within a $\log n$ factor of the optimal. Further approximation results include Eidenbenz [10] who designed algorithms for several variations of terrain guarding problems and Amit *et al.* [11] who analyze heuristics with experimental evidence of good performance in covered area and in the number of guards.

Another approach tackled by Erdem and Sclaroff [12] and Tomás *et al.* in [13] consists of modeling the problem as a discrete combinatorial problem and then solving the corresponding optimization problem. The former discretize the interior of the polygon with a fixed grid, yielding an approximation algorithm and the latter gives empirical analysis of an exact method of successive approximations based on dominance of visibility regions.

Finally, in [1], we presented an exact algorithm to optimally solve the OAGP. In this algorithm, we iteratively discretize and model the problem as a classical *Set Cover problem* (SCP). Besides demonstrating the feasibility of this approach, we showed that, in practice, the number of iterations required to solve instances of up to 200 vertices was very small and that the resulting algorithm turned out to be quite efficient.

Our contribution. Though the number of iterations executed by the exact algorithm we proposed in [1] was shown to be polynomially bounded, its practical performance is much better depending on how the polygon is discretized. This becomes clearer when we notice that at each iteration an instance of SCP, a NP-hard problem, has to be solved at optimality, in our case, by an Integer Programming (IP) solver.

In this paper, we conduct a thorough experimental investigation concerning the trade-off between the number and nature of discretizing points and the number of iterations, analyzing the practical viability of each approach. Our test data, available in [14], includes multiple instances for each size of the vertex set, for various classes of orthogonal polygons with up to *a thousand* vertices.

The new experimental results significantly surpassed those we reported in [1]. This is due to the exploration of alternative discretization strategies, which allow us to address difficult instances as well as to handle a fivefold increase in the polygon size compared to the literature, while attaining low execution times.

Organization of the text. In the next section, we explain the basic ideas that support the algorithm. Section 5.3 is devoted to the description of the algorithm and the alternative strategies to discretize the polygon. Next, in Section 5.4 we give an account of the set up of the testing environment and present the different classes of instances used. Besides, following the recommendations of Johnson [15], McGeoch and Moret [16], Sanders [17] and Moret [18], we show an extensive experimental analysis of the algorithm

implemented with multiple discretization strategies, including the evaluation of multiple measurements. Concluding remarks are drawn in the last section.

5.2 Basics

In an instance of the OAGP we are given an orthogonal simple polygon P that bounds an art gallery and we are asked to determine the minimum number and an optimal placement of vertex guards in order to keep the whole gallery under surveillance. Vertex guards are assumed to have a range of vision of $360\,^{\circ}$.

The approach used by the algorithm described in Section 5.3 transforms the continuous OAGP into a discrete problem which, in turn, can be easily modeled as an instance of the SCP. In fact, for the last two decades, this has been the only known technique for transforming art gallery problems leading to efficient approximation algorithms. Below, we describe in detail the approach used by the algorithm, starting with some basic definitions.

An *n-wall orthogonal art gallery* can be viewed as a planar region whose boundary consists of an orthogonal simple polygon (without holes) P, i.e., one whose n edges are parallel to the x or y axis. The set or vertices of P are denoted by V and a vertex $v \in V$ is called a *reflex vertex* if the internal angle at v is greater then $180\,^{\circ}$. Whenever no confusion arises, *a point in* P will mean a point either in the interior or on the boundary of P.

Any point y is said to be visible from any other point x if and only if the closed segment joining x and y does not intersect the exterior of P. The set $V(v)$ of all points visible from a vertex $v \in V$ is called the *visibility region of* v. In order to determine $V(v)$, we employ the linear time algorithm proposed by Lee [19] and extended by Joe and Simpson [20, 21].

A set of points S is a *guard set* for P if for every point $p \in P$ there exists a point $s \in S$ such that p is visible from s. Hence, a *vertex guard set* G is any subset of vertices such that $\bigcup_{g \in G} V(g) = P$. In other words, a vertex guard set for P gives the positions of stationary guards who can oversee an entire art gallery of boundary P. Thus, the OAGP amounts to finding the smallest subset $G \subset V$ that is a vertex guard set for P.

From the above discussion one can see that the problem of finding the smallest vertex guard set for P can be seen as a *continuous* minimum set cover problem, where every visibility region $V(v), v \in V$ is a set and points $p \in P$ are elements of the set.

Notice that the term *continuous* is used here to denote the fact that there is an infinite number of elements to be covered in the SCP instance, as the points of P in the above definition comprise an infinite set. To cope with this, one can discretize the problem, generating a representative finite number of points in P so that the formulation becomes

manageable. We now describe how the solutions to successively refined discrete instances are guaranteed to converge to an optimal solution to the original continuous problem. To this end, consider an arbitrary discretization of P into a finite set of points $D(P)$. An IP formulation of the corresponding SCP instance is shown below.

$$\begin{aligned} z \;=\; \min \quad & \sum_{j \in V} x_j \\ \text{s.t.} \quad & \sum_{j \in V} a_{ij} x_j \geq 1, \text{ for all } p_i \in D(P) \qquad (1) \\ & x_j \in \{0,1\}, \text{ for all } j \in V \end{aligned}$$

where the binary variable x_j is set to 1 if and only if vertex j from P is chosen to be in the guard set. Moreover, given a point p_i in $D(P)$ and a vertex j of P, a_{ij} is a binary value which is 1 if and only if $p_i \in V(j)$.

Given a feasible solution x for the IP above, let $Z(x) = \{j \in V \mid x_j = 1\}$. Constraint (1) states that each point $p_i \in D(P)$ is visible from at least one selected guard position in the solution and the objective function minimizes the cardinality z of $Z(x)$. Clearly, as the set $D(P)$ is finite, it may happen that $Z(x)$ does not form a vertex guard set for P. In this case, we must add a new point inside each uncovered region and include these points in $D(P)$. A new SCP instance is then created and the IP is solved again.

We are now able to describe the algorithm proposed in [1]. In the *preprocessing phase*, three procedures are executed. The first one computes the visibility polygons for the points in V. The second one computes the initial discretization $D(P)$ and the third one builds the corresponding IP model. In the *solution phase*, the algorithm iterates as described above, solving SCP instances for the current discretization, until no regions remain uncovered.

We had shown in [1] that an upper bound on the number of iterations is $O(n^4)$. This result was derived from the fact that the edges of the visibility regions induce a subdivision of P which is comprised of $O(n^4)$ faces or *Atomic Visibility Polygons* (AVPs). One point inside an AVP is enough to guarantee that this entire AVP will be covered by the solution to the discretized problem. Whence follows the upper bound on the number of iterations. However, it can be derived from a result by Bose *et al.* [22] that $\Theta(n^3)$ is a tight bound on the number of AVPs, improving the aforementioned worst case convergence result.

C1 (Pg.87)

Moreover, the actual number of iterations that is required depends on how many uncovered regions can be successively generated. As the cost of each iteration is related to the number of constraints in (1), an interesting trade-off naturally sprouts and leads one to attempt multiple choices of discretization schemes. On the other hand, any method of cleverly choosing the initial points of the discretization will have a corresponding cost

in preprocessing time, opening another intriguing time exchange consideration. These questions are precisely what we address next.

In Section 5.3 we consider several possible discretization schemes which lead to the various performance analysis discussed in Section 5.4.

5.3 Discretization Strategies

The key point in the IP approach is to set up instances of the set cover problem that can rapidly be solved while minimizing the number of iterations required to attain an optimal solution to the original art gallery problem, within the least amount of time. However, one must take into account that sophisticated geometric properties used to build more efficient discretizations will generate a corresponding cost in preprocessing, possibly outweighing the benefits. Below, we discuss alternatives for the discretization of P.

Regular Grid. The first discretization strategy considered is to generate a dense grid inside the polygon in the assumption that few iterations might be required. This was the main venue for the experimentations described in [1]. Such grid is built with a step of size equal to the smallest gap in the x- and y-coordinates of the vertices of P. We also include all the vertices in this initial discretization.

As it turns out, for some polygons the number of grid points grows quadratically with the number of vertices, inflating the number of constraints in the formulation of the SCP which increases the time needed to solve each instance.

A summary of the outcome of the use of regular grids for two classes of polygons can be seen in Figure 5.3 and Table 5.1.

Induced Grid. Given the perception that reflex vertices are responsible for part of the difficulty of the problem, a natural discretization strategy to be considered is the grid induced by the edge extensions that intersect in the polygon. In this case, we generate fewer constraints than in the previous strategy while attempting to capture more of the intrinsic visibility information of the polygon. One might expect that this could lead to faster to solve instances of set cover while keeping the number of iterations low.

Just Vertices. In one extreme, given that all vertices of the polygon will have to be covered, we consider the rather sparse case where the starting discretization contains just the vertices of the polygon. Initially, this leads to quicker solutions to the set cover problem than the two previous approaches and has the benefit that each additional constraint comes really from "hard to see" regions. Since in this way we avoid any spurious grid points, one might envision that the potentially higher number of iterations could still be compensated by the smaller size of the SCP instances.

Complete Atomic Visibility Polygons. Recall that an AVP of a polygon P is any (convex) face of the subdivision of P induced by the visibility polygons of all its vertices. It then follows that if a guard set G covers the centroid of an AVP, then it must cover the entire AVP. Therefore, if G covers the centroids of every AVP of P, then G must be a guard set for P.

This suggests that we could solve the problem in a single iteration of the algorithm by building an instance of SCP from all these centroids. However, for sizeable instances, this approach would lead to an impractically large instance of up to $O(|V|^3)$ constraints, where V is the set of vertices of P (see [22]).

Nonetheless, as we will see, not all AVPs need to be represented in the set of constraints in order to guarantee a single iteration. Therefore, we do not need to consider this more costly discretization strategy.

Reduced Atomic Visibility Polygons. Following the previous discussion, we now show that we can significantly reduce the number of constraints required to guarantee that the algorithm will find the minimum number of guards necessary to cover P by solving a single instance of the set cover problem.

Firstly, given a vertex $v \in V$, an edge of the visibility polygon $V(v)$ is called a *visibility edge* of v. Furthermore, if it is not an edge of P, then it is called a *proper visibility edge* of v. It follows that an AVP is a face in the arrangement of visibility edges, interior to P. Hence, the edges of an AVP are either portions of edges of P or portions of proper visibility edges of vertices of P. An AVP $\mathcal{V}$ is called a *shadow AVP* if it is not visible from any of the vertices whose proper visibility edges spawn $\mathcal{V}$.

Let $G \subset V$ be a partial guard set for P and let U be a maximal connected region not covered by G. Note that U can be partitioned into a collection of AVPs. To see that at least one of these must be a shadow AVP, notice that if one side of a proper visibility edge of, say, vertex v_i that intersects U, is visible from v_i then the opposite side must not be. Hence, by successive partitioning U, at least one shadow AVP is bound to remain.

The Reduced AVP discretization strategy consists of taking all vertices of P plus the centroids of every shadow AVP. Since any guard set that covers all the points of this discretization cannot leave an uncovered region, it follows that no iterations will be required.

It remains to be experimentally analyzed which of these discretizing strategies will bring about the most benefit, timewise. This is done in the next section.

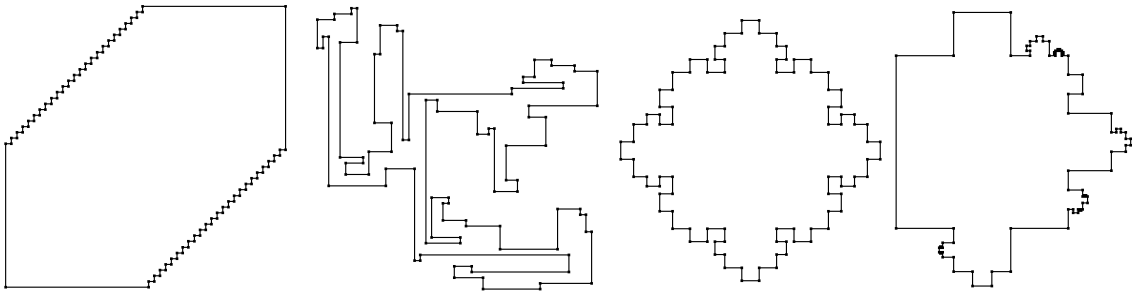


Figure 5.1: Sample polygons with 100 vertices: FAT, Random, Complete von Koch and Random von Koch.

5.4 Computational Experiments

We now present an experimental evaluation of the several discretization strategies discussed in the previous section. We coded all variants of the algorithm described in earlier sections along with a visibility algorithm from [20]. The implementation was done in `C++`, compiled with `GNU g++ 4.1`, on top of `CGAL 3.2.1`, and used the IP solver `Xpress v17.01.02`. As for hardware, we used a desktop PC featuring a Pentium IV at 3.4 GHz and 1 GB of RAM running `GNU/Linux 2.6.17`.

5.4.1 Instances

We conducted the tests on a large number of instances downloadable from [14] and grouped into four different classes (see Figure 5.1). The first two of these classes are composed of n-vertex orthogonal polygons placed on an $n/2 \times n/2$ unit square grid and devoid of collinear edges, as suggested in [23] and the last two are based on a modified version of the von Koch curve (see [24]).

(1) **FAT:** This class was introduced in [13] as an extreme scenario for the IP approach and also used in [1], where instances with up to 200 vertices were solved to optimality.

(2) **Random:** These are n-vertex randomly generated orthogonal polygons created using the algorithm proposed in [23].

C2 (Pg.87)

(3) **Complete von Koch (CvK):** These polygons were generated based on a modified version of the von Koch curve. The fractal has a Hausdorff dimension of 1.34 and is generated, starting with a square, by recursively replacing each edge as shown in Figure 5.2, where $\overline{ar} = \overline{st} = \overline{ub}$ and $\overline{sr} = \overline{tu} = \frac{3}{4}\overline{ar}$.

(4) **Random von Koch (RvK):** This class consists of randomized von Koch instances of up to level 4. Starting from a square, each of these instances is generated iteratively until the desired number of vertices is reached. In each iteration, we randomly choose an edge of the current polygon, with level smaller than 4, and decide in a random fashion

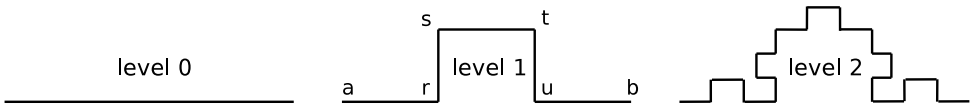


Figure 5.2: Levels of modified von Koch polygons.

whether we expand it or not.

The FAT and Random instances were generated for the number of vertices n in the ranges: [20, 200] with step 20, (200, 500] with step 50 and (500, 1000] with step 100. Similar sizes were chosen for the RvK class. The CvK class contains by construction only 3 instances with $n \in \{20, 100, 500\}$.

For our conclusions to be endowed with statistical significance, we had to decide on the sample size (number of instances generated), for each value of n, in the classes Random and RvK. To this end, we ran our algorithm on random instances, while varying the sample size s. We concluded that the variance of the results remains practically unchanged after $s \geq 30$ and, therefore, we decided to generate 30 instances for each value of n. It is worth noting that, up to scaling, only one instance is defined for a given n in the FAT class, hence no decision on sample size is needed in this case.

Thus, in total, our data set is composed of 1833 OAGP instances, having between 20 and 1000 vertices, i.e., our largest instances are five times the largest ones whose optimal solutions are reported in the literature.

5.4.2 Results

We now discuss the experimental evaluation of the different strategies described in Section 5.3. All values reported here are average results for 30 instances of each size, or 30 runs of the same instance, for FAT and CvK classes.

C3 (Pg.87)

The FAT instances were introduced in [13] as an extremal scenario for the IP approach because of the larger number of constraints resulting from regular discretizations of P. Figure 5.3 displays the amount of time spent by the exact algorithm on the FAT class with each discretization strategy. It can be seen that there is a huge difference between the strategies, though all the discretizations lead to a solution in only one iteration. Notice that, in this case, the Regular and Induced Grids coincide, leading to the same running times. On the other hand, the *Reduced AVP* and *Just Vertices* discretizations are both composed of only the vertices of P, since FAT polygons have no shadow AVPs. Of course, the *Reduced AVP* strategy spent more time on the preprocessing phase, which causes the difference seen in the chart. However, the two strategies can deal with FAT polygons with up to 1000 vertices in reasonable time, going far beyond the results reported earlier for this class which are limited to 200-vertex polygons.

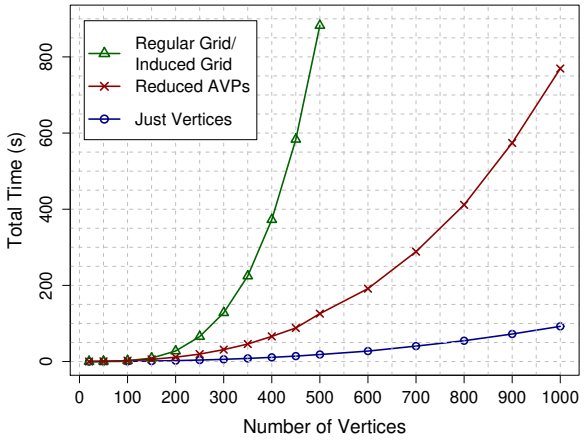


Figure 5.3: Total time: FAT polygons.

Table 5.1: Complete von Koch polygons.

# vertices	Final $\|D(P)\|$			Total Time (s)		
	20	100	500	20	100	500
Reg. Grid	45	500	6905	0.05	1.57	92.37
Ind. Grid	24	205	1665	0.03	1.41	70.94
Red. AVPs	28	324	5437	0.07	3.14	143.93
Just Vert.	20	107	564	0.04	0.97	29.35

The usage of discretization strategies based on dense grids becomes more discouraging when we analyze the results in Table 5.1. This table displays the execution time and the size of the discretization of the strategies proposed in Section 5.3 for the CvK polygons. One can see that for these instances, the *Induced Grid* strategy has a better performance than the *Regular Grid* strategy. The size of the discretization produced by *Regular Grid* grows quadratically in the number of vertices, and thus inflates the number of constraints in the IP formulation increasing considerably the time necessary to optimally solve the SCP instances. The *Reduced AVP* strategy has a poor behavior for CvK polygons since the number of shadow AVPs increases fast in this case. The *Just vertices* strategy is again the one that spends less time.

Figure 5.4 shows the amount of discretized points necessary for each strategy to achieve the optimal solution of OAGP for Random (in (a)) and RvK (in (b)) polygons. Especially from the Random case, one can see that the *Regular Grid* strategy rapidly becomes impractical due to the huge size of the discretization and, therefore, will no longer be analyzed for other classes of polygons. On the other hand, one can see that the *Reduced AVP* strategy still follows the same behavior of the CvK case for RvK instances, with the discretization size growing fast as the number of vertices of P increases. Nevertheless, this approach is very well-suited for random polygons. The curves corresponding to the *Just Vertices* strategy suggest that the set of vertices of the polygon is a good guess for the initial discretization since few new points are added to it to achieve the optimal solution of an OAGP instance for both classes of instances.

Figure 5.5 shows the number of iterations each strategy needs to achieve the optimal solution for both classes of random polygons. The chart in (a) displays the expected behavior with the number of iterations increasing as the size of the discretizations decrease.

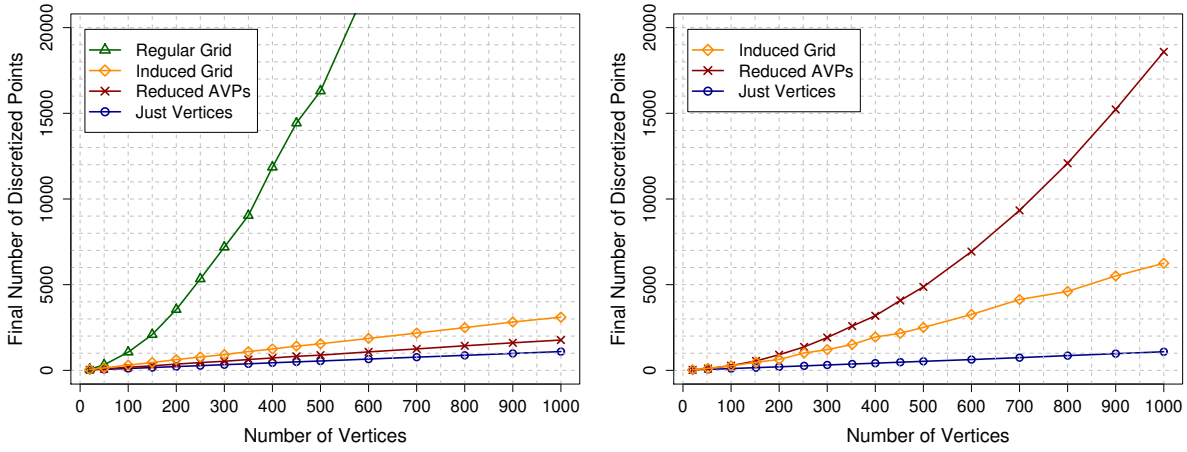


Figure 5.4: Final discretization size: (a) Random polygons; (b) Random von Koch polygons.

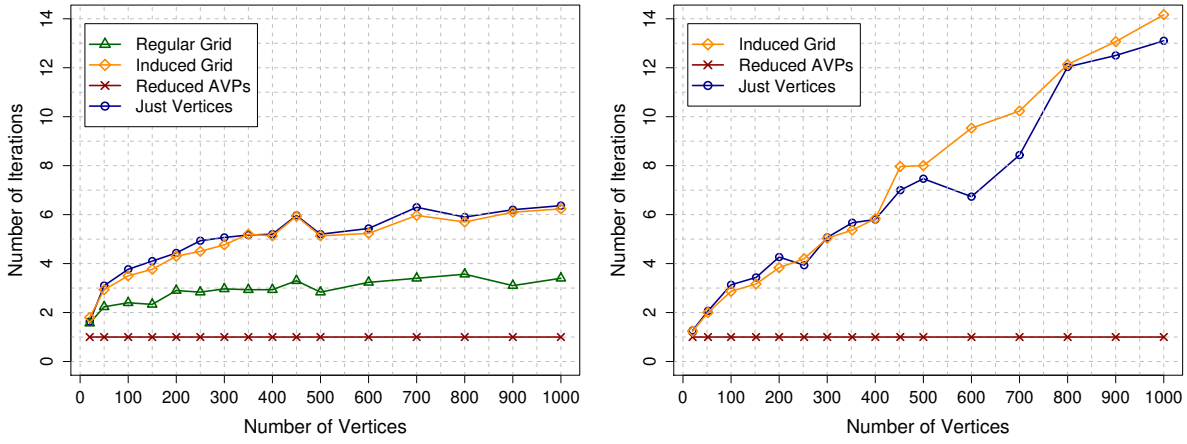


Figure 5.5: Number of iterations: (a) Random polygons; (b) Random von Koch polygons.

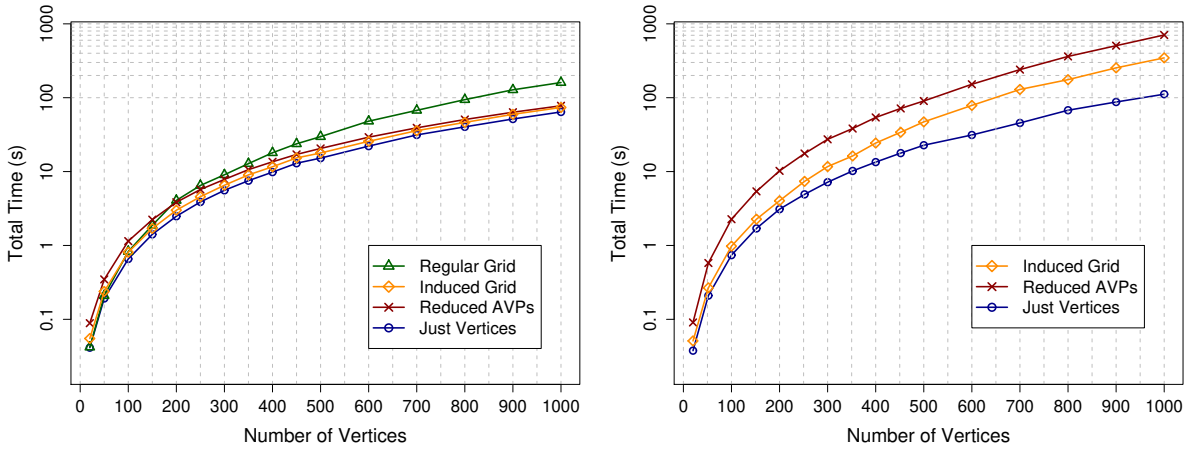


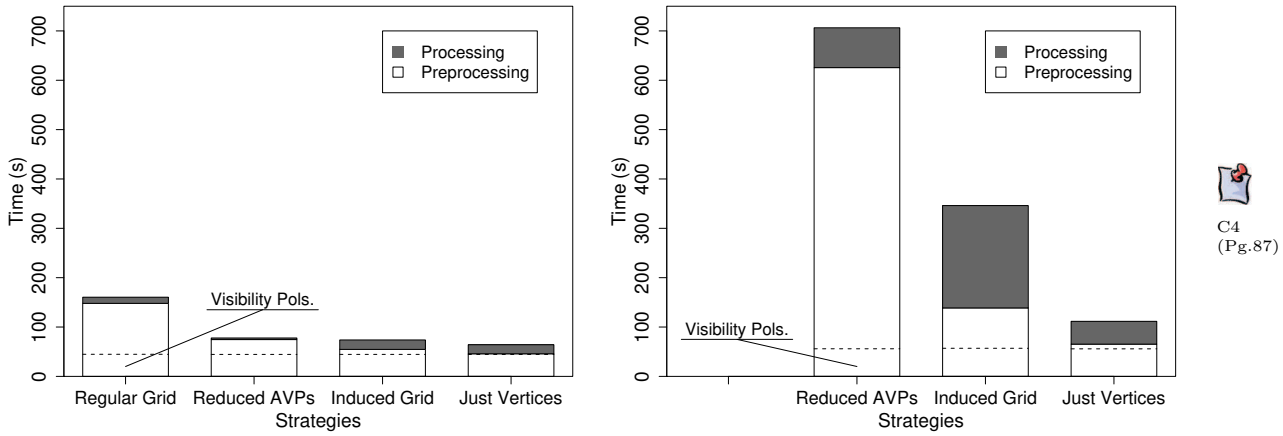
Figure 5.6: Total time: (a) Random polygons; (b) Random von Koch polygons.

Now, relative to the size of the input polygon, the number of iterations remains negligible when compared to the theoretical bound of $\Theta(n^3)$. In chart (b) relative to RvK polygons, the number of iterations increases a bit faster with the instance size but is still small. Somewhat surprisingly, in this case *Induced Grid* iterates slightly more than the *Just vertices* strategy.

Figure 5.6 shows the total amount of time, including the preprocessing and processing phases, to solve instances from the random classes. Notice that the curves are plotted in log ×linear format and both charts are in the same scale. One can see that for Random polygons, all the strategies behave similarly except, as expected and explained before, the *Regular Grid*. The tendency of *Just vertices* strategy is very similar in both classes of polygons. This shows that, though we are solving harder instances in the RvK case, the strategy is robust.

We now turn our attention to the time spent by the algorithm in each phase for the discretization strategies. Recall that the preprocessing phase is composed of three procedures. The first one is common to all strategies and computes the visibility polygons. The second one computes the initial discretization and its cost is highly affected by the choice of the strategy to be implemented. The worst case corresponds to the *Reduced AVP* strategy since it requires the computation of all AVPs and the determination of the shadow AVPs and of their centroids. On the other extreme, we have the *Just vertices* strategy where no computation is needed. Finally, in the third procedure of the preprocessing phase one has to build the starting IP model and the time spent in doing so depends on the size of the discretization, which again benefits the *Just vertices* strategy.

In Figure 5.7 one can see that the time spent in the preprocessing phase is in accordance with the discussion above, the *Reduced AVP* strategy being the most time consuming for



C4
(Pg.87)

Figure 5.7: Execution time for polygons of 1000 vertices: (a) Random polygons; (b) Random von Koch polygons. The lower part of the preprocessing time corresponds to the construction of the visibility polygons.

RvK. What is somehow surprising is that, though we are solving NP-hard problems in the solution phase, the majority of the time consumption refers to the preprocessing phase, which is entirely polynomial. The extraordinary developments in IP solvers together with the fact the SCP instances arising from OAGP are among the easy ones can explain this apparently counter intuitive behavior of the algorithm. Thus, for the *Reduced AVP* strategy to become competitive, a cleverer and faster procedure has to be developed to discard not only shadow AVPs but other ones. Comparing the size of the final discretizations of the different strategies shown earlier there seems to be room for such improvements.

5.5 Conclusions and Remarks

In this paper, we conducted an experimental investigation of an exact algorithm for the Orthogonal Art Gallery problem (OAGP) proposed in [1] which relies on the discretization of the interior of the input polygon P and on the modeling of this simplified discrete problem as a Set Cover problem (SCP). The resulting SCP instance is solved to optimality by an IP solver and, if uncovered regions remain, additional constraints are included and the process is repeated. Clearly, the performance of the algorithm depends on the number of such iterations.

This work focused on different strategies to implement the discretization of P. Thorough experimentation was carried out to assess the trade-off between the number of iterations and time spent by the many variants of the algorithm that arise from the alternative discretization methods.

Table 5.2: Total Time (seconds): *Just Vertices* strategy.

	Polygons Classes			
n	Random	FAT	RvK	CvK
100	0.65	0.58	0.73	0.97
500	15.21	18.47	22.73	29.35
1000	64.13	92.41	111.55	♯

C5
(Pg.88)

Our conclusion is that this exact algorithm is a viable choice to tackle instances of the OAGP, in light of the fact that the largest ones we solved were five times larger than those reported earlier in the literature.

The apparent advantage of a discretization which ensures an exact solution after a single iteration of the algorithm (like the *Reduced AVP* strategy) did not prove to be effective in practice. This became even clearer when we compared its results with those of the *Just Vertices* strategy which, represents the opposite extreme situation, as, in principle, it starts with the smallest "natural" SCP instance. However, as one can see from Table 5.2 this strategy leads to a very fast implementation that takes only few seconds of CPU time to solve OAGP instances with up to 1000 vertices.

The success of this exact algorithm clearly benefits from the extraordinary developments in IP solvers in recent years, which lead to the solution of large instances of SCP in a very small amount of time. Therefore, we believe that the *Reduced AVP* strategy can only become competitive with the *Just Vertices* strategy when the preprocessing time required by the former is significantly reduced. Though we used powerful data structures and packages to perform the necessary geometrical operations, we could not significantly lessen the preprocessing time which, for the largest instances tested here, correspond roughly to the time required by the IP solver to resolve ten instances of SCP.

A promising venue of further investigation lies in trying to identify inexpensive geometric properties that might lead to a set of constraints that capture the essence of the hardness of the problem, such as a significant reduction on the number of AVPs.

References

[1] Couto, M.C., de Souza, C.C., de Rezende, P.J.: An exact and efficient algorithm for the orthogonal art gallery problem. In: Proc. of the XX Brazilian Symp. on Comp. Graphics and Image Processing, IEEE Computer Society (2007) 87–94

[2] Honsberger, R.: Mathematical Gems II. Number 2 in The Dolciani Mathematical Expositions. Mathematical Association of America (1976)

[3] Chvátal, V.: A combinatorial theorem in plane geometry. Journal of Combinatorial Theory Series B **18** (1975) 39–41

[4] Urrutia, J.: Art gallery and illumination problems. In Sack, J.R., Urrutia, J., eds.: Handbook of Computational Geometry. North-Holland (2000) 973–1027

[5] Kahn, J., Klawe, M.M., Kleitman, D.: Traditional galleries require fewer watchmen. SIAM J. Algebraic Discrete Methods **4** (1983) 194–206

[6] Schuchardt, D., Hecker, H.D.: Two NP-hard art-gallery problems for ortho-polygons. Mathematical Logic Quarterly **41** (1995) 261–267

[7] Sack, J.R., Toussaint, G.T.: Guard placement in rectilinear polygons. In Toussaint, G.T., ed.: Computational Morphology. North-Holland (1988) 153–175

[8] Edelsbrunner, H., O'Rourke, J., Welzl, E.: Stationing guards in rectilinear art galleries. Comput. Vision Graph. Image Process. **27** (1984) 167–176

[9] Ghosh, S.K.: Approximation algorithms for art gallery problems. In: Proc. Canadian Inform. Process. Soc. Congress. (1987)

[10] Eidenbenz, S.: Approximation algorithms for terrain guarding. Inf. Process. Lett. **82**(2) (2002) 99–105

[11] Amit, Y., Mitchell, J.S.B., Packer, E.: Locating guards for visibility coverage of polygons. In: Proc. Workshop on Algorithm Eng. and Experiments. (2007) 1–15

[12] Erdem, U.M., Sclaroff, S.: Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. Comput. Vis. Image Underst. **103**(3) (2006) 156–169

[13] Tomás, A.P., Bajuelos, A.L., Marques, F.: On visibility problems in the plane - solving minimum vertex guard problems by successive approximations. In: Proc. of the 9th Int. Symp. on Artificial Intelligence and Mathematics. (2006)

[14] Couto, M.C., de Souza, C.C., de Rezende, P.J.: OAGPLIB - Orthogonal art gallery problem library. *www.ic.unicamp.br/∼cid/Problem-instances/Art-Gallery*.

[15] Johnson, D.S.: A theoretician's guide to the experimental analysis of algorithms. In et al., M.H.G., ed.: Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implem. Challenges. AMS, Providence (2002) 215–250

[16] McGeoch, C.C., Moret, B.M.E.: How to present a paper on experimental work with algorithms. SIGACT News **30** (1999)

[17] Sanders, P. In: Presenting data from experiments in algorithmics. Springer-Verlag New York, Inc., New York, NY, USA (2002) 181–196

[18] Moret, B.: Towards a discipline of experimental algorithmics. In: Proc. 5th DIMACS Challenge

[19] Lee, D.T.: Visibility of a simple polygon. Comput. Vision, Graphics, and Image Process. **22** (1983) 207–221

[20] Joe, B., Simpson, R.B.: Visibility of a simple polygon from a point. Report CS-85-38, Dept. Math. Comput. Sci., Drexel Univ., Philadelphia, PA (1985)

[21] Joe, B., Simpson, R.B.: Correction to Lee's visibility polygon algorithm. BIT **27** (1987) 458–473

[22] Bose, P., Lubiw, A., Munro, J.I.: Efficient visibility queries in simple polygons. Computational Geometry **23**(3) (2002) 313–335

[23] Tomás, A.P., Bajuelos, A.L.: Generating random orthogonal polygons. In: Current Topics in Artificial Intelligence. Volume 3040 of LNCS., Springer (2004) 364–373

[24] Falconer, K.: Fractal Geometry, Mathematical Foundations and Applications. John Wiley & Sons (1990) pp. 120–121.

Comentários

C1: Limite Teórico do Algoritmo

Para mais detalhes, ver Comentário 1 na página 68.

C2: Polígonos de von Koch

Os polígonos de von Koch utilizados nessa dissertação estão descritos em detalhes no Capítulo 2, Seção 2.4 e são baseados numa versão modificada da curva de von Koch, definida em [25]. Como todo fractal, possui uma dimensão de Hausdorff [28] que excede a dimensão da sua topologia. A dimensão de Hausdorff é calculada pela seguinte equação:

$$D = \frac{\ln N}{\ln r}$$

onde, N pode ser considerado como o número de objetos similares ao original que foram criados quando este foi dividido em r partes.

C3: Massa de Testes

Para o experimento possuir um embasamento estatístico, é necessário que antes seja feito um estudo sobre o número de instâncias e execuções que devem ser geradas para cada tipo de instância e cada valor n de vértices desejado. Detalhes deste estudo estão descritos no Capítulo 2, Seção 2.4.

C4: Composição do Tempo de Execução do Algoritmo

O tempo total utilizado pelo algoritmo é composto pelo tempo de preprocessamento, onde é feito o cálculo das regiões de visibilidade dos candidatos a guardas, a construção da discretização inicial, a geração do modelo PLI e também a carga desse modelo no resolvedor inteiro utilizado; e pelo tempo de processamento, onde são resolvidas as instâncias do problema NP-DIFÍCIL (SCP), calculam-se as áreas não cobertas pela solução atual, encontram-se os novos pontos para refinar a discretização e, por fim, constróem-se e adicionam-se as novas restrições ao modelo PLI. Note que, experimentalmente, o tempo de preprocessamento excede, em muito, o tempo de processamento. Embora o preprocessamento tenha seu tempo limitado polinomialmente, seguindo a discussão do Comentário 5 da página 69, tem-se que, para as instâncias de teste, o resolvedor inteiro utilizado encontra uma solução para cada instância do problema NP-DIFÍCIL em um tempo ínfimo.

C5: Ausência do CvK de 1000 vértices

Os polígonos completos de von Koch são gerados pela expansão de um fractal. Cada nível expandido eleva o número de vértices em uma quantidade determinada ($n = 4 \times 5^l$, onde l é o nível desejado). Isto faz com que não exista a instância com 1000 vértices e como a próxima, com 2500 vértices, estava muito além da maior instância resolvida até o momento da escrita daquele artigo, ela não foi incluída nesses experimentos.

Capítulo 6

An Exact Algorithm for an Art Gallery Problem

Prólogo

Neste capítulo, apresenta-se o texto submetido para publicação em um periódico internacional arbitrado. Esse trabalho foi escrito em co-autoria com os professores Pedro J. de Rezende e Cid C. de Souza, ambos do Instituto de Computação da Universidade Estadual de Campinas, onde uma versão preliminar foi publicada como relatório técnico no final de 2009 [11]. Ele reflete o estado da pesquisa na fase final desta dissertação, onde trabalha-se com polígonos simples sem obstáculos, além dos polígonos ortogonais já apresentados. É também feito um estudo de novas alternativas para a discretização inicial da galeria e o tamanho das instâncias de teste foi aumentado em duas vezes e meio, se comparadas com o trabalho anterior. Dois grandes objetivos são almejados com esse trabalho. O primeiro é instituir que o algoritmo não resolve apenas instâncias ortogonais, mas também as instâncias simples sem obstáculos sem praticamente nenhuma alteração. O segundo, é utilizar novas alternativas para a discretização inicial e, com isso, aumentar o tamanho das instâncias para validar experimentalmente essas estratégias.

Abstract

We consider an Art Gallery problem (AGP) which aims to minimize the number of vertex guards required to monitor an art gallery whose boundary is an n-vertex simple polygon. In this paper, we compile and extend our research on exact approaches for solving the AGP. In prior works [1, 2], we proposed and tested an exact algorithm for the case of orthogonal polygons. In that algorithm, a discretization that approximates the polygon is used to formulate an instance of the Set Cover Problem which is subsequently solved to optimality. Either the set of guards that characterizes this solution solves the original instance of the AGP, and the algorithm halts, or the discretization is refined and a new iteration begins. This procedure always converges to an optimal solution of the AGP and, moreover, the number of iterations executed highly depends on the way we discretize the polygon. Notwithstanding that the best known theoretical bound for convergence is $\Theta(n^3)$ iterations, our experiments show that an optimal solution is always found within a small number of them, even for random polygons of many hundreds of vertices. Herein, we broaden the family of polygon classes to which the algorithm is applied by including non orthogonal polygons. Furthermore, we propose new discretization strategies leading to additional trade-off analysis of preprocessing vs. processing times and achieving, in the case of the novel *Convex Vertices* strategy, the most efficient overall performance so far. We report on experiments with both simple and orthogonal polygons of up to 2500 vertices showing that, in all cases, no more than 15 minutes are needed to reach an exact solution, on a standard desktop computer. Ultimately, we more than doubled the size of the largest instances solved to optimality compared to our previous experiments, which were already five times larger than those previously reported in the literature.

Keywords: art gallery problem, exact algorithm, set covering, visibility.

6.1 Introduction

According to Honsberger [5], in 1973, Victor Klee posed to Vasek Chvátal the question of determining the minimum number of watchmen sufficient to guard an art gallery shaped as an n-wall simple polygon. Chvátal's proof [6] that $\lfloor n/3 \rfloor$ guards are occasionally necessary and always sufficient for that purpose was the first result in what has become a whole new field of study. Witnessing to the broad spectrum of literature that has appeared since, we have O'Rourke's classical book [7], Shermer's and Urrutia's surveys [8, 9] and a multitude of journal papers cited within these. More recently, Ghosh's book [10], which presents a vast set of topics on visibility problems spun from related questions, cites over three hundred references, mostly from the last fifteen years.

Among the earliest and most important results, one finds Lee and Lin's NP-completeness proof of a related minimization problem [11] that remained open for more than ten years. Namely, given a planar simple polygon P, determine a placement of a *minimum* number of stationary guards that cover P.

Many variants of this problem have been considered in the literature. The formulation we study here restricts the placement of guards to vertices of the polygon that represents the outer boundary of a given art gallery. Throughout this paper, we will refer to this particular formulation as the *Art Gallery Problem* (AGP). Furthermore, we consider general simple polygons as well as the subclass of simple orthogonal polygons, which are particularly relevant due to most real life buildings and galleries being orthogonally shaped [9].

One of the earliest major result concerning the latter problem, due to Kahn *et al.* [12], states that $\lfloor n/4 \rfloor$ guards are occasionally necessary and always sufficient to cover an orthogonal polygon with n vertices. Later, Schuchardt and Hecker proved that minimizing the number of guards in this variation is also NP-hard [13], settling a question that remained open for almost a decade [14].

Several placement algorithms have been proposed in the past, such as Edelsbrunner *et al.* [15] and Sack and Toussaint [14], which deal with the problem of efficiently placing exactly $\lfloor n/4 \rfloor$ guards covering a given orthogonal gallery.

On the other hand, in a recently published paper [17], based on [16], Ghosh presents an $O(n^4)$ time approximation algorithm for simple polygons yielding solutions within a $\log n$ factor of the optimal. Further approximation results include Eidenbenz [18] who designed algorithms for several variations of terrain guarding problems and Amit *et al.* [19] who analyze heuristics with experimental evidence of good performance in covered area as well as in the number of guards.

Another approach tackled by Erdem and Sclaroff [20] and Tomás *et al.* in [21] consists of modeling the problem as a discrete combinatorial problem and then solving the corresponding optimization problem. The former discretizes the interior of the polygon with a fixed grid, yielding an approximation algorithm and the latter gives empirical analysis of an exact method of successive approximations based on dominance of visibility regions.

Finally, in [1], we presented an exact algorithm to optimally solve the orthogonal AGP. In this algorithm, we iteratively discretize and model the problem as a classical *Set Cover problem* (SCP) [30]. Besides demonstrating the feasibility of this approach, we showed that, in practice, the number of iterations required to solve instances of up to 200 vertices was very small and that the resulting algorithm turned out to be quite efficient.

Though the number of iterations executed by the exact algorithm we proposed in [1] and improved in [2] was shown to be polynomially bounded, its practical performance is far better depending on how the polygon is discretized (see also [3] or watch [4]). This

becomes evident when we notice that in each iteration an instance of SCP, an NP-hard problem, has to be solved to optimality, in our case, by an Integer Programming solver.

In this paper, we build upon all our previous studies and conduct a thorough experimental investigation concerning the trade-off between the number and nature of discretizing methods and the number of iterations and we analyze the practical viability of each approach. Moreover, while our previous works [1, 2] dealt only with orthogonal polygons, here we show that the same approach works well for general simple polygons. Besides dealing with new classes of polygons, two new and superior discretization strategies are introduced in this paper and are compared to the previously studied ones. All of our test data are available in [22] and include multiple instances for each size of the vertex set, for various classes of polygons with up to 2500 vertices.

The new experimental results significantly surpass those we reported in [1, 2]. This is due to the exploration of alternative discretization strategies, which allow us to address difficult instances as well as to handle a substantial increase in polygon size compared to earlier results, while still attaining low execution times.

In the next section, we describe the process of modeling the AGP as an SCP and the basic ideas necessary for the description of the algorithm which appears in Section 6.3, along with its proof of correctness and complexity. Section 6.4 is devoted to the description of the alternative strategies to discretize the input polygon. Next, in Section 6.5 we give an account of the set up of the testing environment and present the different classes of instances used. Complying with the recommendations of Johnson [23], McGeoch and Moret [24], Sanders [25] and Moret [26], we show in Section 6.5.2 an extensive experimental analysis of the algorithm considering multiple discretization strategies, and include an evaluation of various comparative measurements. Concluding remarks are drawn in the last section.

6.2 Modeling

In an instance of the AGP, we are given a simple polygon that bounds an art gallery and we are asked to determine the minimum number and an optimal placement of vertex guards in order to keep the whole gallery under surveillance. Vertex guards are assumed to have a range of vision of 360°.

The approach used by the algorithm described in Section 6.3 transforms the continuous AGP into a discrete problem which, in turn, can easily be modeled as an instance of the SCP. In fact, for the last two decades, this has been the only known technique for developing efficient approximation algorithms for the art gallery problem [17, 18, 19]. Before we present our algorithm and establish its correctness, let us review some basic definitions.

An *n-wall art gallery* can be viewed as a planar region whose boundary consists of a simple polygon (without holes) P. The set of vertices of P is denoted V and a vertex $v \in V$ is called a *reflex vertex* if the internal angle at v is greater then 180°. Whenever no confusion arises, *a point in* P will mean a point either in the interior or on the boundary of P.

Any point y in P is said to be visible from any other point x in P if and only if the closed segment joining x and y does not intersect the exterior of P. The set $\text{Vis}(v)$ of all points in P visible from a vertex $v \in V$ is called the *visibility region of* v. It is easy to see that $\text{Vis}(v)$ is always a star shaped polygon. A boundary description of $\text{Vis}(v)$ can be computed in linear time by an algorithm proposed by Lee [27] and extended by Joe and Simpson [28, 29].

A set of points S is a *guard set* for P if for every point $p \in P$ there exists a point $s \in S$ such that p is visible from s. Hence, a *vertex guard set* G is any subset of vertices such that $\bigcup_{g \in G} \text{Vis}(g) = P$. In other words, a vertex guard set for P gives the positions of stationary guards who can oversee an entire art gallery of boundary P. Thus, an AGP amounts to finding the smallest subset $G \subset V$ that is a vertex guard set for P.

The reader who is familiar with the set cover problem (SCP) [30] must already have perceived that the problem of finding the smallest vertex guard set for P can be regarded as a specific SCP. Namely, we wish to find a smallest cardinality set of star-shaped polygons (visibility regions of the vertices of P) whose union cover P. Notice that, strictly speaking, this is a *continuous* SCP since there are infinitely many points in the interior of P to be covered. However, one can discretize the problem by generating a finite number of representative points in P so that the formulation becomes manageable. We shall see below how this approach will lead us to a viable solution of the original problem.

Incidently, we should also cite that an LP-based approach to estimate bounds for the number of guards required to monitor a polygonal art gallery has recently been attempted in [31], even though no guarantee of termination or of exactness is presented.

We now describe how the solutions to successively refined discrete instances are guaranteed to converge to an optimal solution to the continuous problem. To this end, consider an arbitrary discretization of P into a finite set of points $D(P)$. We will denote by $I(P, D(P))$ an instance of the discretized Art Gallery problem generated in this fashion. An IP formulation of the corresponding SCP instance is shown below.

$$
\begin{aligned}
z \;=\; \min \quad & \sum_{j \in V} x_j \\
\text{s.t.} \quad & \sum_{j \in V} a_{ij} x_j \geq 1, \text{ for all } p_i \in D(P) \qquad (1) \\
& x_j \in \{0, 1\}, \text{ for all } j \in V
\end{aligned}
$$

where the binary variable x_j is set to 1 if and only if vertex j of P is chosen to be in the guard set. Moreover, given a point p_i in $D(P)$ and a vertex j of P, a_{ij} is a binary value which is 1 if and only if $p_i \in \text{Vis}(j)$.

Given a feasible solution x to the IP above, let $Z(x) = \{j \in V \mid x_j = 1\}$. Constraint (1) states that each point $p_i \in D(P)$ is visible from at least one selected guard position in the solution and the objective function minimizes the cardinality z of $Z(x)$. Clearly, as the set $D(P)$ is finite, it may happen that $Z(x)$ does not form a vertex guard set for P. In this case, we must pick a new point inside each uncovered region and include these points in $D(P)$. A new SCP instance is then created and the corresponding IP is solved, leading to an iterative procedure. At the end of Section 6.3.2 we establish the convergence of this process.

The actual number of iterations that are required depends on how many uncovered regions might be successively generated. As the cost of each iteration is related to the number of constraints in (1), an interesting trade-off naturally sprouts and leads one to attempt multiple choices of discretization schemes. On the other hand, any method of cleverly choosing the initial points of the discretization will have a corresponding cost in preprocessing time, opening another intriguing time exchange consideration. These questions are precisely what we address in Section 6.4 where we consider several possible discretization schemes which lead to the various performance analysis discussed in Section 6.5.

6.3 Description of the algorithm and proof of correctness

The algorithm is divided into two phases: a *Preprocessing Phase*, where the initial discretization described in Section 6.1 is constructed and the Integer Programming problem is set up, and a *Solution Phase* in which the algorithm iterates as described above, solving SCP instances for the current discretization, until no regions remain uncovered.

As mentioned earlier, a solution set $Z(x)$ to the discretized formulation in Section 6.2 may not always constitute a guard set for P since there might be regions inside P that are not visible from any guard in $Z(x)$.

To formalize our subsequent reasoning, we start with the following definition.

Definition 6.3.1. *Let $I(P, D(P))$ be an instance of the discretized Art Gallery problem with polygon P as the gallery boundary and $D(P)$ a discretization of P. A solution $Z(x)$ of this instance is called* viable *if $Z(x)$ is a guard set for P, i.e.,*

$$\bigcup_{g \in Z(x)} Vis(g) = P.$$

Any exact method for the original AGP which solves its discretized version must address the fact that a solution to $I(P, D(P))$ might not necessarily be viable. As we will see, our algorithm overcomes this difficulty and always produces a viable solution by successively refining the given discretization whenever it detects that the present solution is not viable. Furthermore, the following theorem establishes that a solution obtained through this iterative process is also minimal.

Theorem 6.3.1. *Let Z be a solution of an instance $I(P, D(P))$ of the discretized Art Gallery problem. If Z is viable then Z is optimal.*

Proof. From the fact that Z is a solution of the minimization problem $I(P, D(P))$, it follows that Z is optimal as a vertex guard cover for the set $D(P)$ of points which discretize the polygon P, i.e., $z = |Z|$ is minimum among the cardinalities of all vertex guard covers of $D(P)$.

Now, let Z^* be an optimal vertex guard set for P and let $z^* = |Z^*|$. Since Z^* is also a vertex guard cover for $D(P)$, we must have $z^* \geq z$. On the other hand, since Z^* is viable, it follows that $z \geq z^*$. □

Theorem 6.3.1 establishes that when the algorithm finds a solution for the discretized formulation which is viable, that solution is also a minimum vertex guard cover for P, i.e., it is a guard set for P.

We are now able to describe in detail the algorithm we first proposed in [1]. In the *Preprocessing Phase*, three procedures are executed: the first one computes the visibility polygons for the points in V, the second one computes the initial discretization $D(P)$ and the third one builds the corresponding IP model. In the *Solution Phase*, the discretized problem is successively solved and refined until a viable (and optimal) solution is found.

6.3.1 Preprocessing Phase

The main steps of the preprocessing phase are summarized in Algorithm 6.3.1.

In order to assemble the formulation outlined in Section 6.2, we start by building an initial discretization $D(P)$ of the polygon (step 1). In Section 6.4 we describe alternative discretization strategies and their impact on the efficiency of this algorithm.

Once a discretization is built, we compute which of its points are located inside the visibility region of each vertex in V, and, then, include these restrictions in the SCP formulation.

The total complexity of step 3 is $O(n^2)$ [28] and, assuming that $m = |D(P)|$, the full complexity of step 5 is $O(nm \log n)$ since point location of each of the m points of $D(P)$ in a star-shaped visibility n-polygon can be accomplished in $O(\log n)$ time. Hence, the overall complexity of the preprocessing phase is dominated by that of step 5 whenever $m \in \Omega(n/\log n)$, and by that of step 3, otherwise.

Algorithm 6.3.1 Preprocessing Phase

1: $D(P) \leftarrow$ chosen initial discretization of P;
2: **for** each $j \in V$ **do**
3: Compute Vis(j);
4: **for** each discretization point $p_i \in D(P)$ **do**
5: $a_{ij} \leftarrow$ **Boolean**($p_i \in$ Vis(j));
6: **end for**
7: **end for**

The result of the preprocessing phase is an Integer Programming (IP) formulation for the Set Cover problem which, once solved, generates a solution Z that, while not necessarily constituting a guard set for P, will always cover all the points in $D(P)$.

6.3.2 Solution Phase

In the second phase of the algorithm, starting from the IP formulation generated in the preprocessing phase, we solve the discretized instance followed by an iterative refinement of the discretization until the solution becomes viable. This refinement is attained by generating one more point in the discretization for each uncovered region (e.g., its centroid) and by adding the corresponding constraints to the current SCP. These additional points enhance the formulation and lead to a solution closer to a viable one. Algorithm 6.3.2 outlines the steps executed in the solution phase.

Algorithm 6.3.2 Solution Phase

1: **repeat**
2: $Z \leftarrow$ solution of $I(P, D(P))$;
3: **for** each uncovered region R **do**
4: $c \leftarrow$ centroid of R;
5: $D(P) \leftarrow D(P) \cup \{c\}$;
6: Add a new row, r, to the set of constraints (1) corresponding to point c:
$\sum_{j \in V} a_{rj} x_j \geq 1$ where $a_{rj} \leftarrow$ **Boolean**($c \in$ Vis(j)), $\forall j \in V$;
7: **end for**
8: **until** Z is viable

It remains to be argued that Algorithm 6.3.2 halts, as it will then follow from Theorem 6.3.1 that the algorithm is exact and the solution given is indeed a guard set for P.

Definition 6.3.2. *Consider the set of all visibility regions of the vertices in V, whose union, obviously, covers P. The edges of these visibility regions induce an arrangement*

of line segments within P whose faces we call atomic visibility polygons*, or AVPs (see Figure 6.1).*

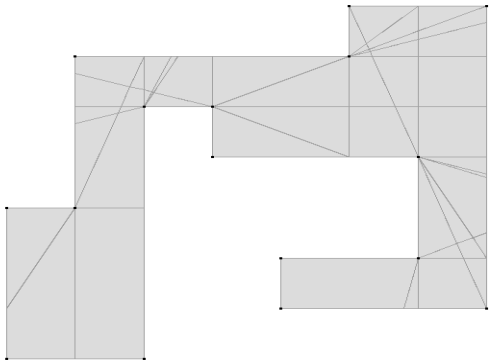


Figure 6.1: Visibility arrangement and AVPs.

In order to determine the worst case for the number of iterations executed by the algorithm, firstly, note that it follows from the definition of the AVPs that if the centroid of (or, for that matter, any point in the interior of) an atomic visibility polygon $\mathcal{V}$ is visible from a vertex guard, the entire area of $\mathcal{V}$ must also be.

As the visibility region of any vertex can have at most $O(n)$ edges, the induced arrangement is generated from $O(n^2)$ line segments and has a total complexity of no more than $O(n^4)$ faces (or AVPs).

Note that in step 3, any uncovered region (witness to the fact that Z is not viable) is necessarily a simple polygon formed by the union of neighboring AVPs. Therefore, an upper bound on the maximum number of iterations effected by the algorithm is $O(n^4)$ and this establishes the convergence.

Moreover, it can be derived from a result by Bose *et al.* [32] that $\Theta(n^3)$ is a tight bound on the number of AVPs, improving the above worst case result. However, in practice, this is still hugely over estimated and should be regarded solely as proof of convergence of the iterative method.

6.4 Discretization strategies

As presented in the previous section, the convergence of the algorithm follows from an upper bound on the number of uncovered regions. Yet, as each iteration solves an instance of an NP-hard problem (the set cover problem, SCP), the chosen discretization strategy must ideally be light enough to set up instances of SCP that can quickly be solved while minimizing the number of iterations required to attain an optimal solution.

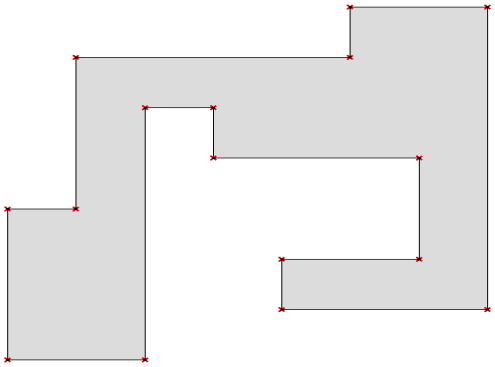


Figure 6.2: Example of the initial discretization used in the *All Vertices* strategy.

Thus, there is a tradeoff between speed and precision that must be taken into account when designing a good discretization strategy. While the use of sophisticated geometric properties to build more efficient discretizations may reduce the number of iterations done by the algorithm, the corresponding cost in preprocessing might outweigh its benefits.

The following sections go into details on several alternatives for the discretization of the polygon and discusses the theoretical advantages and possible drawbacks of each one.

6.4.1 Single vertex

The simplest strategy one might consider is to start a discretization with a single vertex of the polygon P. At first glance, the *Single Vertex* strategy may seem weak since a single point in P conveys no geometric information and the solution of the discretized AGP associated to this simple discretization is expected to leave several uncovered regions, leading to a number of iterations of the algorithm.

However, the size of the SCP instances that the *Single Vertex* strategy generates is very small and they come without preprocessing cost. Therefore, it is still worth determining whether this strategy pays off.

6.4.2 All vertices

A reasonable approach to try to reduce the number of iterations from the *Single Vertex* strategy is to start with a larger discretization whose points are adequately distributed over P. However, to maintain the benefits of the previous strategy, the number of points in such discretization should be kept small and easy to compute. The *All Vertices* strategy is an attempt to reach this goal. We consider the still sparse case where the starting discretization contains all the vertices of the polygon (see Figure 6.2).

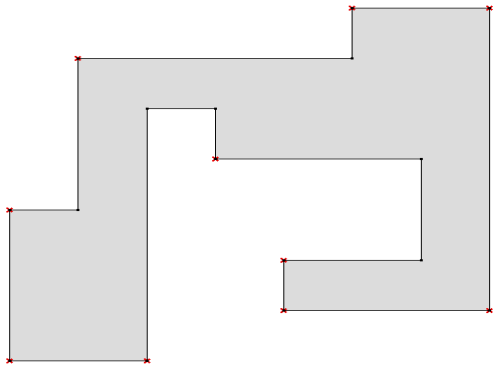


Figure 6.3: Example of the initial discretization used in the *Convex Vertices* strategy.

One can see that this strategy explores the fact that the vertices of the polygon should capture enough geometric information to prevent uncovered regions near the convex vertices from emerging.

Furthermore, experiments show that the *All Vertices* strategy generates smaller uncovered regions than the *Single Vertex* strategy and, in this case, more meaningful constraints get added, leading to better solutions in each iteration.

6.4.3 Convex vertices

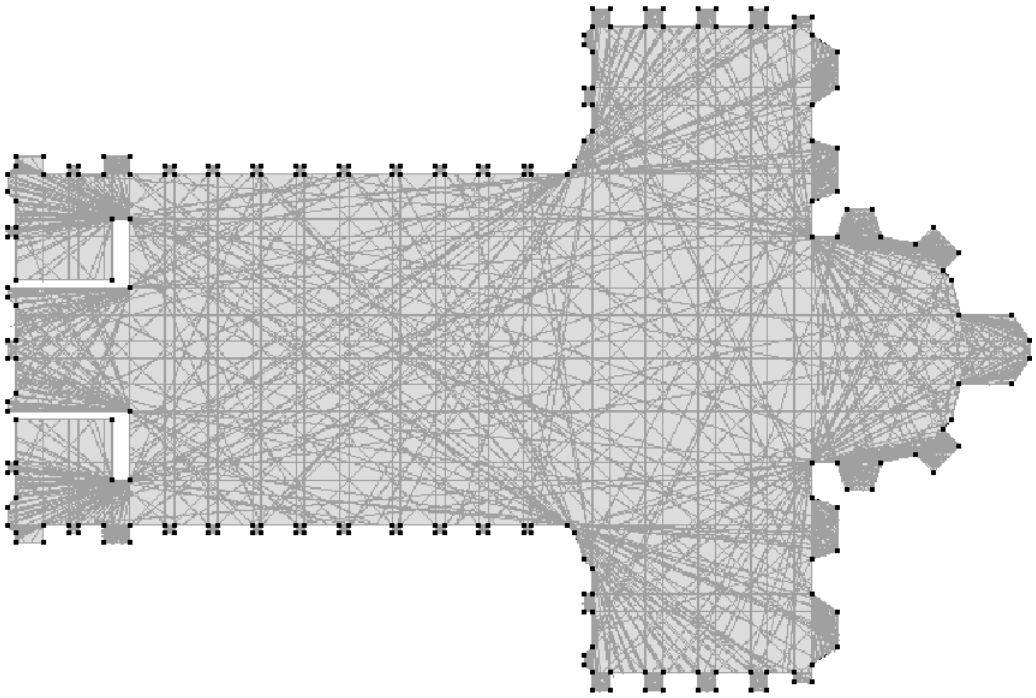
Convex vertices are clearly more useful discretization points than reflex vertices since these are more easily guarded than those. Therefore, if any vertices might be redundant in gathering visibility information, it is natural to assume that the reflex ones are the most superfluous.

These observations lead us to consider the *Convex Vertices* strategy which starts with a discretization of P composed solely by the convex vertices (see Figure 6.3). In doing this, we further reduce discretization size, while still capturing much of the combinatorial visibility relationships at the price of a negligible increase in preprocessing cost.

While one might expect that this reduction could increase the number of iterations, we detected no such consequence. Besides, this strategy preserves the same nice features of the two previous alternatives, namely, an inexpensive preprocessing phase and SCP instances with a low number of constraints in each iteration.

6.4.4 AVPs

The strategies seen so far seek to keep the number of constraints in the initial SCP instance small, while trying to reduce the number of iterations. However, it is possible to devise a strategy that can lower the number of iterations to one.



C1
(Pg.117)

Figure 6.4: A sizeable gallery and its visibility arrangement showing all AVPs.

To this end, one has to identify a set of regions that, when covered, guarantee that the whole gallery is guarded. Furthermore, these regions must have the property that, once one of its points is visible from a vertex-guard, the entire region is watched by that guard. Once these regions are discerned, a discretization can be built by picking one point in each of them.

As shown in the previous section, the atomic visibility polygons (AVPs) of P fulfill both properties stated in the above paragraph. Therefore, the initial discretization containing the centroids of all AVPs leads to an SCP instance that, once solved, produces an optimal vertex-guard set for P. However, as there can be $O(n^3)$ AVPs, the number of constraints in the SCP model might also be $O(n^3)$.

Although this proves that we could solve the problem in a single iteration of the algorithm, building a discretization with the centroids of all AVPs of a complex gallery could result in a huge SCP instance (see Figure 6.4). Nonetheless, as shown next, not all AVPs need to be represented in the set of constraints in order to guarantee a single iteration, which makes the *all AVPs* discretization pointless.

Shadow AVPs.

As seen before, solving an SCP instance with the centroids of all AVPs is very costly. However, we can significantly reduced the number of discretized points and still guarantee

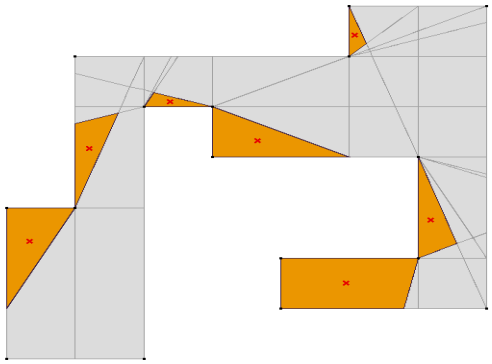


Figure 6.5: Example of the initial discretization used in the *Shadow AVPs* strategy.

that the algorithm finds the minimum number of guards necessary to cover P after the first iteration. In order to do so, we introduce the notion of a *shadow* AVP.

Initially we say that a line segment is a *visibility edge* for P if there exists a vertex $v \in P$ such that this segment is an edge of $\text{Vis}(v)$. Moreover, a visibility edge e originated from vertex v is said to be *proper* for v if and only if e is not contained in any edges of P.

Notice that since an AVP is a face in the arrangement generated by the visibility edges, the edges of an AVP are either portions of edges of P or portions of proper visibility edges of vertices of P.

We say that an AVP $\mathcal{S}$ is a *shadow AVP* if there exists a subset U of vertices of V such that $\mathcal{S}$ is not visible from any vertex in U and the only proper visibility edges that spawn $\mathcal{S}$ emanate from vertices in U.

The *Shadow AVPs* discretization strategy consists of taking the centroids of every shadow AVP (see Figure 6.5).

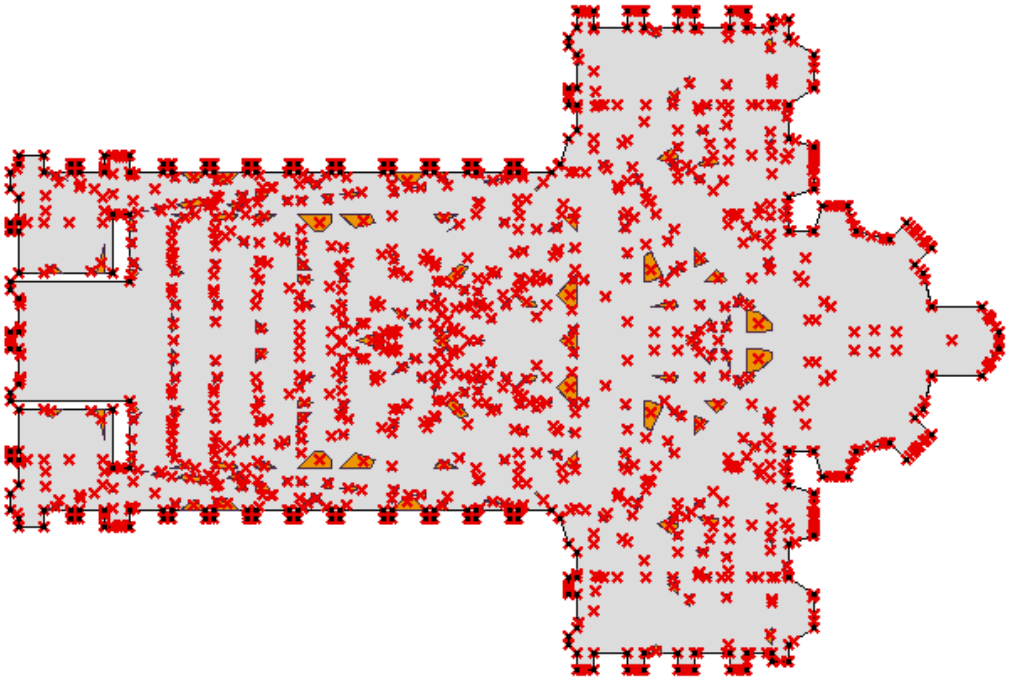
We now establish the fundamental relation between the optimal solutions of the discretized AGP with the *Shadow AVPs* strategy and those of the original AGP.

Theorem 6.4.1. *Let $I(P, D(P))$ be an instance of the discretized Art Gallery problem for polygon P where $D(P)$ is the set of centroids of the shadow AVPs of P. Then, G is a vertex-guard set for $D(P)$ if and only if G is a vertex-guard set for P.*

Proof. The necessity part is trivial since $D(P) \subset P$, therefore, we focus on the proof of sufficiency.

Suppose $G \subset V$ is a vertex-guard set for $D(P)$, but not for P. Thus, there exist regions of P that are not covered by any of the vertices of G. Let R be a maximal connected region not covered by G. Note that R is the union of (disjoint) AVPs.

To prove that at least one of those AVPs is of type shadow, notice that the entire region R is not seen by any vertex in G whose proper visibility edges spawn R. If R is an AVP, it is by definition a shadow AVP. Otherwise, there is a vertex $v_i \in V$ which



C1
(Pg.117)

Figure 6.6: A sizeable gallery and the discretization points used in the *Shadow AVPs* strategy.

has a proper visibility edge e_{vi} that intersects and partitions R in two other regions. One of these regions matches the side of e_{vi} visible from v_i while the opposite one does not. Hence, through an inductive argument, by successively partitioning R, at least one shadow AVP is bound to be contained in R and therefore uncovered.

This contradicts the hypothesis since G is a guard set for $D(P)$ which is comprised of the centroids of all shadow AVPs. □

From Theorem 6.4.1, it is clear that with the *Shadow AVPs* strategy the algorithm converges in one iteration. Also, when we restrict ourselves to shadow AVPs, the size of the discretization decreases considerably when compared to the AVP strategy. Even for complex galleries (contrast Figures 6.4 and 6.6), this reduction may be large enough to render the algorithm practical.

Although the *Shadow AVPs* strategy requires only a single iteration of the algorithm to find an optimal solution, we will see later that the time spent in the preprocessing phase may become the bottleneck of the algorithm. This issue is investigated in Section 6.5.2.

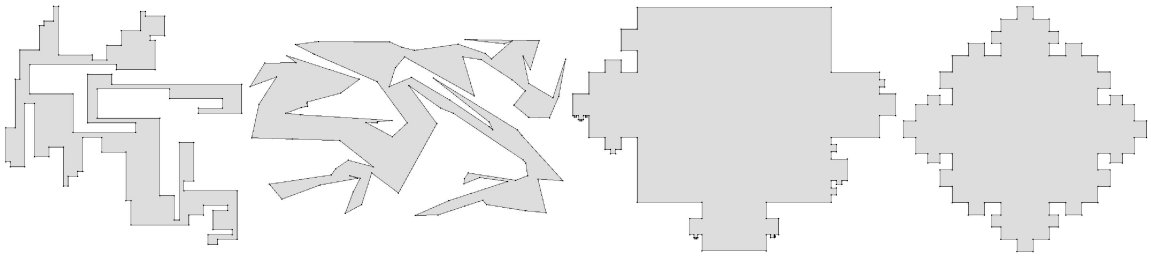


Figure 6.7: Sample polygons with 100 vertices: Random Orthogonal, Random Simple, Random von Koch and Complete von Koch.

6.4.5 Other Strategies

For completeness, it should be mentioned that other strategies have been considered in our prior works, such as those based on the regular [1] and on the induced [2] grid discretizations. Experiments have shown that none of them are competitive with the strategies discussed in this section, which are far more efficient.

C2 (Pg.117)

6.5 Computational Experiments

We now discuss the experimental investigations that we carried out to evaluate the algorithm proposed in Section 6.3. In particular, we analyze the behavior of the algorithm with respect to the various discretization strategies discussed in the previous section.

All our programs were coded in `C++` and compiled with `GNU g++ 4.2`, on top of `CGAL 3.2.1` [33]. The visibility algorithm from [28] was implemented and `Xpress v18.10.04` [34] was used to compute the IP models corresponding to the SCP instances.

As for hardware, we used a desktop PC featuring a Pentium IV at 3.4 GHz and 3 GB of RAM running `GNU/Linux 2.6.24`. We observe that no other processes were allowed to execute in the machine during our tests. Besides, for each instance, the algorithm stopped either because an optimal solution was found or because the program ran out of memory.

6.5.1 Instances

To be considered a realistic method to solve the AGP, an algorithm must be able to handle a large variety of instances with distinctive characteristics. Thus, to test the robustness of our algorithm, we devised four classes of instances (see Figure 6.7). Each of them captures peculiar geometric properties that either appear in actual art galleries or represent extreme situations needed to exercise some of the algorithm's characteristics. The set of all instances used in the experiments reported here, plus several others, are downloadable at [22].

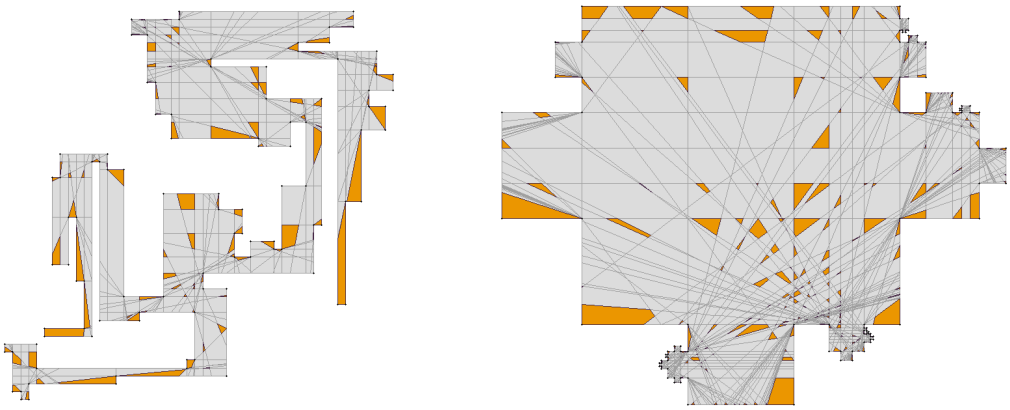


Figure 6.8: Instances of Random Orthogonal and Random von Koch polygons with the same size but distinct complexities.

In the first two classes of instances, the art gallery is represented as an orthogonal or as a simple polygon, respectively. The former are thought to be good representatives of many actual art gallery buildings. The last two classes correspond to polygons assembled from a closed version of the von Koch fractal curve (see [35]). Instances generated in this way tend to have small protuberances on the boundary of the polygon which create tiny areas that are visible only by a small number of vertices. These instances are supposedly harder to solve than similarly sized instances of the two first classes. As an example, consider the two instances in Figure 6.8. Both polygons have 100 vertices, but the visibility arrangement of the Random Orthogonal instance has 2216 edges and 1085 AVPs (with only 99 Shadow AVPs) while the one corresponding to the Random von Koch polygon has 8794 edges and 4420 AVPs (with 264 Shadow AVPs).
More details on how to generate the test polygons of each class are given below.
(1) **Random Orthogonal:** A Random Orthogonal instance consists of a random n-vertex orthogonal polygon placed on an $\frac{n}{2} \times \frac{n}{2}$ unit square grid. The polygon is generated devoid of collinear edges in accordance to the method described in [36].
(2) **Random Simple:** Each Random Simple instance amounts to a random simple polygon generated by a special purpose procedure available in `CGAL` [33]. Essentially, this procedure starts by distributing the vertices of the polygon uniformly in a given rectangle and applies the method of elimination of self-intersections using 2-opt moves.
(3) **Complete von Koch (CvK):** Here, polygons are generated based on a modified version of the von Koch curve, which is a fractal with Hausdorff dimension of 1.34. An instance is created by starting with a square and recursively replacing each edge by five other edges as shown in Figure 6.9, where $\overline{ar} = \overline{st} = \overline{ub}$ and $\overline{sr} = \overline{tu} = \frac{3}{4}\overline{ar}$.

Let us make use of Figure 6.9 to introduce some notation needed to describe the last class of instances. We say that an edge of the current polygon which remains over the

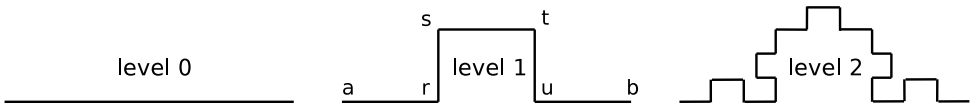


Figure 6.9: Levels of von Koch polygons.

boundary of the initial square is at level 0. When the replacement operation, illustrated in the figure, is applied to an edge e at level k, the new edges that are not collinear with e are said to be at level $k + 1$.

(4) **Random von Koch (RvK):** An instance of this last class is constructed as follows. We start with a square and iteratively apply the replacement operation (from Figure 6.9) to some edges until the number of vertices of the polygon reaches an *a priori* fixed limit. At each iteration, we select an edge at random whose level is smaller than a given parameter λ and randomly decide whether to replace it or not.

It is important to remark that for Random von Koch class, the instances of up to 1000 vertices were generated with λ set to 4. Beyond this size, since the number of vertices nears that of the Complete von Koch polygon of level 4 (2500), λ was set to 5. This is a likely explanation for the discontinuity observed around 1000 vertices in certain plots shown in Section 6.5.2 (see Figure 6.13) where results obtained by our algorithm for this class of polygons are displayed.

The random instances were generated for the number of vertices, n, in the ranges: $[20, 100]$ with step size 20; $(100, 1000]$ with step size 100 and $(1000, 2500]$ with step size 250. Similar sizes were chosen for the RvK class. Lastly, the CvK class contains, by construction, only 4 instances with 20, 100, 500 and 2500 vertices.

To endow our conclusions with statistical significance, we had to define the sample size, i.e, the number of instances generated for each value of n for the classes Random Orthogonal, Random Simple and RvK. To this end, we analyzed the variance of the results produced by our algorithm while we changed the sample size s. We observed that the variance remained practically unchanged for $s \geq 30$ and, therefore, we decided to generate 30 instances for each value of n.

Therefore, in total, our data set is composed of 1804 instances, having between 20 and 2500 vertices each. It is worth noting that our largest instances more than double the size of the largest ones whose optimal solutions are reported in the literature.

For completeness, we mention that other classes of instances were also considered in our prior works [1, 2], including the FAT polygons, introduced in [21]. These classes were not included in the research presented here because they are by far less challenging than the ones considered in the experiments reported in this paper. As an example, FAT polygons of any size always admit an analytical optimal solution consisting of only two guards.

Table 6.1: Results for complete von Koch polygons.

	Final discretization size				Total Time (in secs)			
# of vertices	20	100	500	2500	20	100	500	2500
Single Vertex	9	82	326	1185	0.04	1.23	26.26	808.95
Convex Vertices	12	60	279	1403	0.04	0.84	21.09	720.99
All Vertices	20	115	552	2687	0.03	1.22	27.50	828.54
Shadow AVPs	20	244	5029	-	0.06	2.43	143.75	-


C3
(Pg.117)

Next section presents an extensive experimental analysis of the algorithm considering multiple discretization strategies.

6.5.2 Results

We now discuss the experimental evaluation of the various discretization strategies described in Section 6.4. All values reported in this section are average results for 30 instances of each size, or 30 runs of the same instance in the case of the CvK class, since for this class there is a single instance of each size.

Recall that the discretization size determines the number of constraints in the SCP instance solved in the second step of Algorithm 6.3.2. We start by analyzing the relationship between the discretization types and the time spent by the proposed algorithm. CvK polygons are particularly suited to this analysis because they illustrate two extreme situations. In strategies *Single Vertex*, *Convex Vertices* and *All Vertices* the initial discretizations correspond to very sparse grids whose sizes increase only by a small factor throughout the iterations. On the other hand, for *Shadow AVPs* a single iteration of the algorithm suffices, at the expense of building an extremely dense grid. Table 6.1 summarizes these results.

Notice that *Single Vertex*, *Convex Vertices* and *All Vertices* strategies indeed produce small discretizations whose sizes increase linearly in the number of vertices of P. As for the *Shadow AVPs* strategy, the size of the discretization grows dramatically fast, to the point that we were not able to solve the largest CvK instance on account of running into memory limitations. Large grids inflate the number of constraints in the IP formulation, considerably increasing the time necessary to optimally solve the SCP instance. The *Convex Vertices* strategy is the one that spends less time, followed by *Single Vertex* and *All Vertices*. As it can be seen, the relative order among these strategies remains unchanged with respect to the sizes of the final discretizations.

On the other hand, Figure 6.10 shows the number of discretized points required by each strategy to achieve an optimal solution of the AGP for *Random Orthogonal*, *Random Simple* and *Random von Koch* polygons. One can see that the *Shadow AVPs* strategy

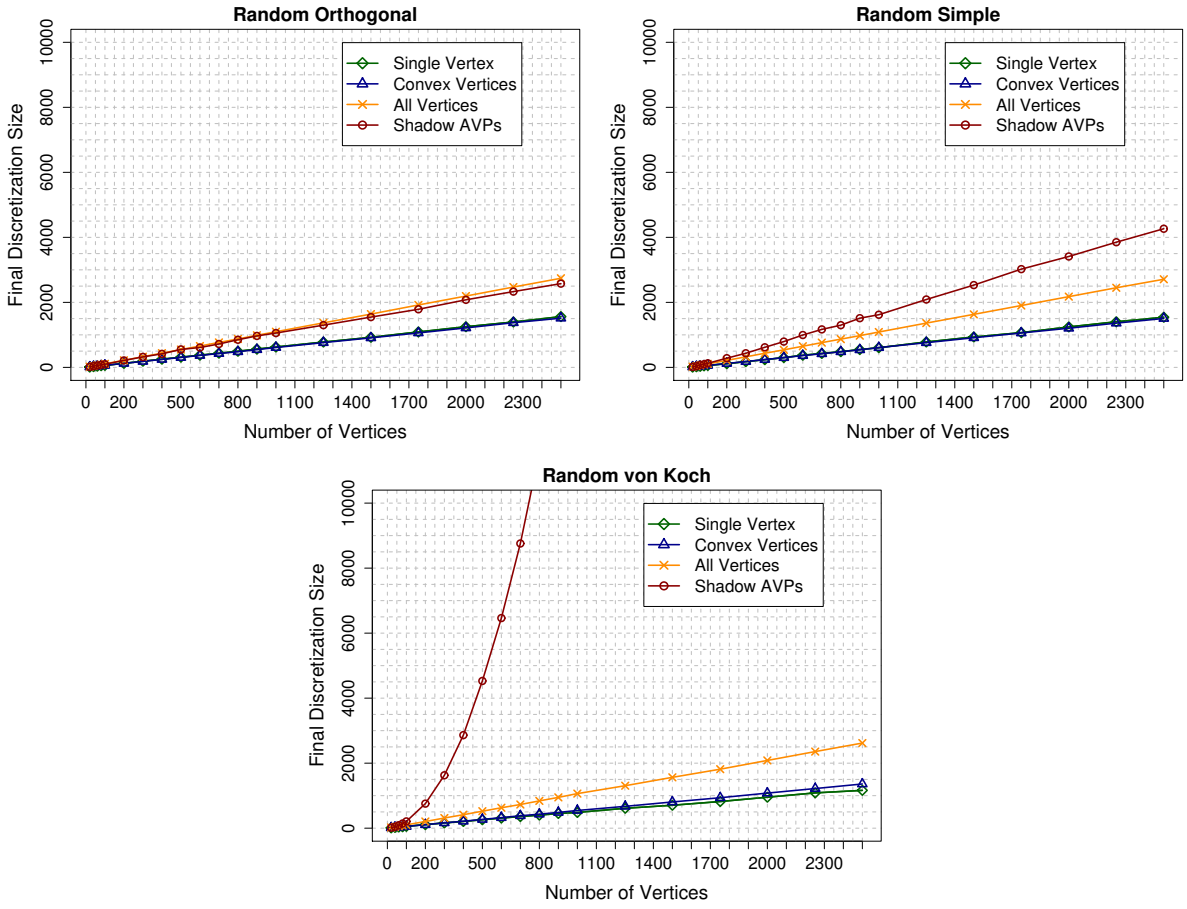


Figure 6.10: Final discretization size by polygon type.

follows the same pattern of the CvK case for RvK instances, with the discretization size growing quickly as the number of vertices of the polygons increases. Nevertheless, as we will see, the *Shadow AVPs* approach is well-suited for random polygons. The curves corresponding to the *All Vertices* strategy suggest that the set of vertices of the polygon is a good bid for the initial discretization since few new points are added to it to achieve an optimal solution of an AGP instance in all three classes under consideration. Notice that the same observation applies to the strategies *Single Vertex* and *Convex Vertices*. Hence, one can infer that small well-chosen proper subsets of V might suffice to capture a relevant part of the hardness of the problem. However, as we will soon see, strategies starting from minuscule discretizations, the extreme case being represented by *Single Vertex*, may cause the algorithm to iterate too much, increasing the computation time.

Figure 6.11 displays the number of iterations each strategy requires to reach an optimal solution for the three classes of random polygons. The lines corresponding to the *Shadow*

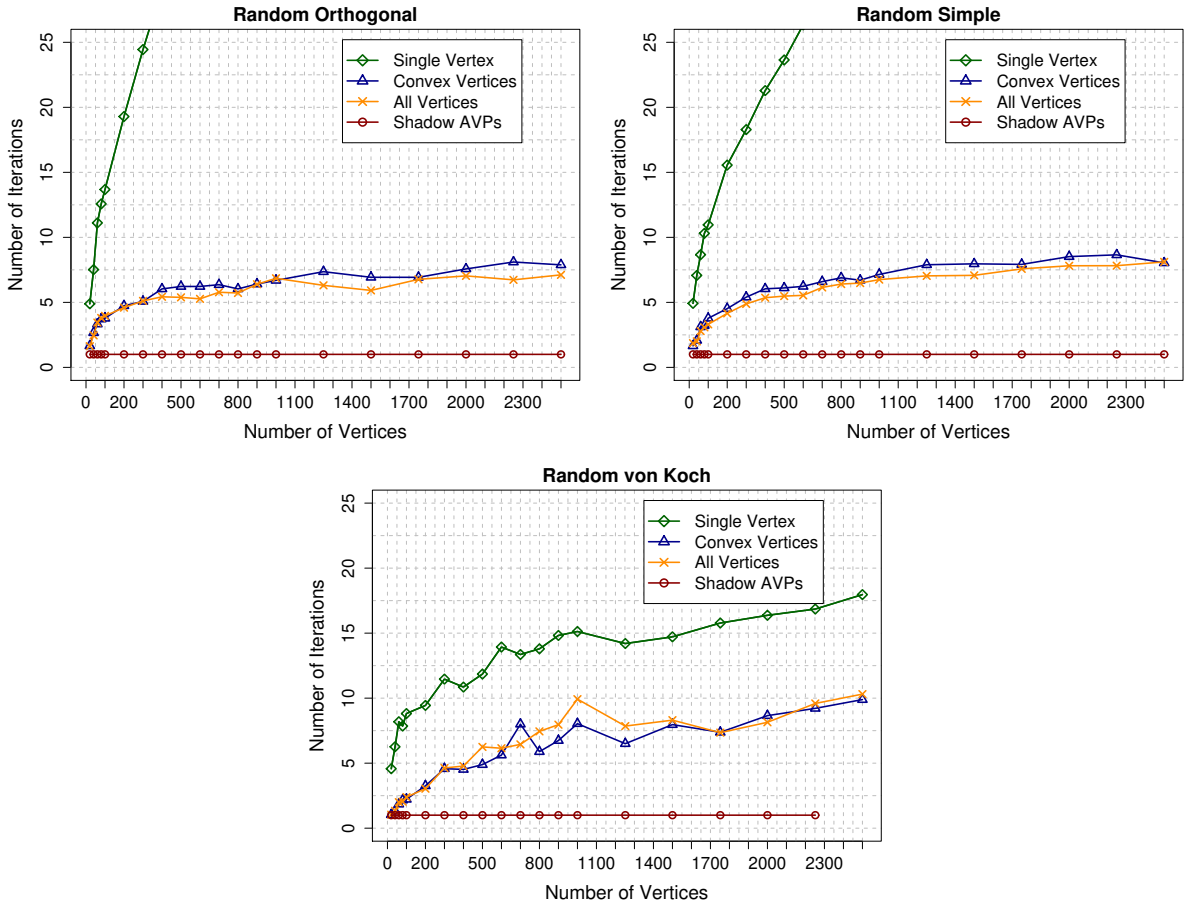


Figure 6.11: Number of iterations by polygon type.

AVPs strategy equate the constant function of value one and are included in the graphs solely as a reference. We recall that we successfully ran our program for the RvK class for instances of only up to 2250 vertices, due to memory limitations.

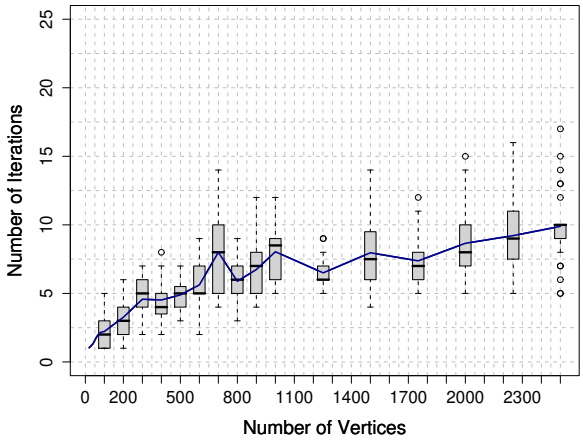
C3 (Pg.117)

Consider now the *Single Vertex* strategy. For *Random Orthogonal* and *Random Simple* polygons, any single point of P only captures strictly local information about the shape of the polygon. Thus, by starting with a unitary discretization, several iterations should be expected before $D(P)$ is dense enough to capture the shape of P. This situation is very clearly depicted in the *Single Vertex* curves for *Random Orthogonal* and *Random Simple* classes. As for the RvK instances, the visibility polygon of most vertices corresponds to large portions of P, leading to many multiply covered areas within P. So, even when we start with a singleton, convergence happens much faster than with the other two classes.

Now, looking at the three non constant curves within each graph, we see that the number of iterations increases as the size of the *initial* discretizations decreases. In ref-

erence to the size of the input polygon, the number of iterations remains negligible when compared to the theoretical bound of $\Theta(n^3)$ (see Section 6.3.2). In regard to RvK polygons, the number of iterations grows a bit faster with the instance size but it still stays quite small. On the other hand, the proximity of the curves for *Convex Vertices* and *All Vertices* shows that the convex vertices alone seems to capture the shape P well enough to dispense with the reflex vertices. Therefore, the *Convex Vertices* strategy iterates just slightly more than the *All Vertices* strategy.

Figure 6.12 exhibits a box-plot chart with the number of iterations performed by the algorithm using the *Convex Vertices* strategy for the RvK class. For most instance sizes, one can see that the average and median values are very close. The difference between the upper and lower quartiles never exceeds 5 and, in all but the largest instances, no more than one outlier occurs. Even for the 2500-sized polygons, whose behavior seems unusual, the total of 8 outliers out of the 30 instances is quite acceptable. This confirms the robustness of the proposed algorithm with this strategy.

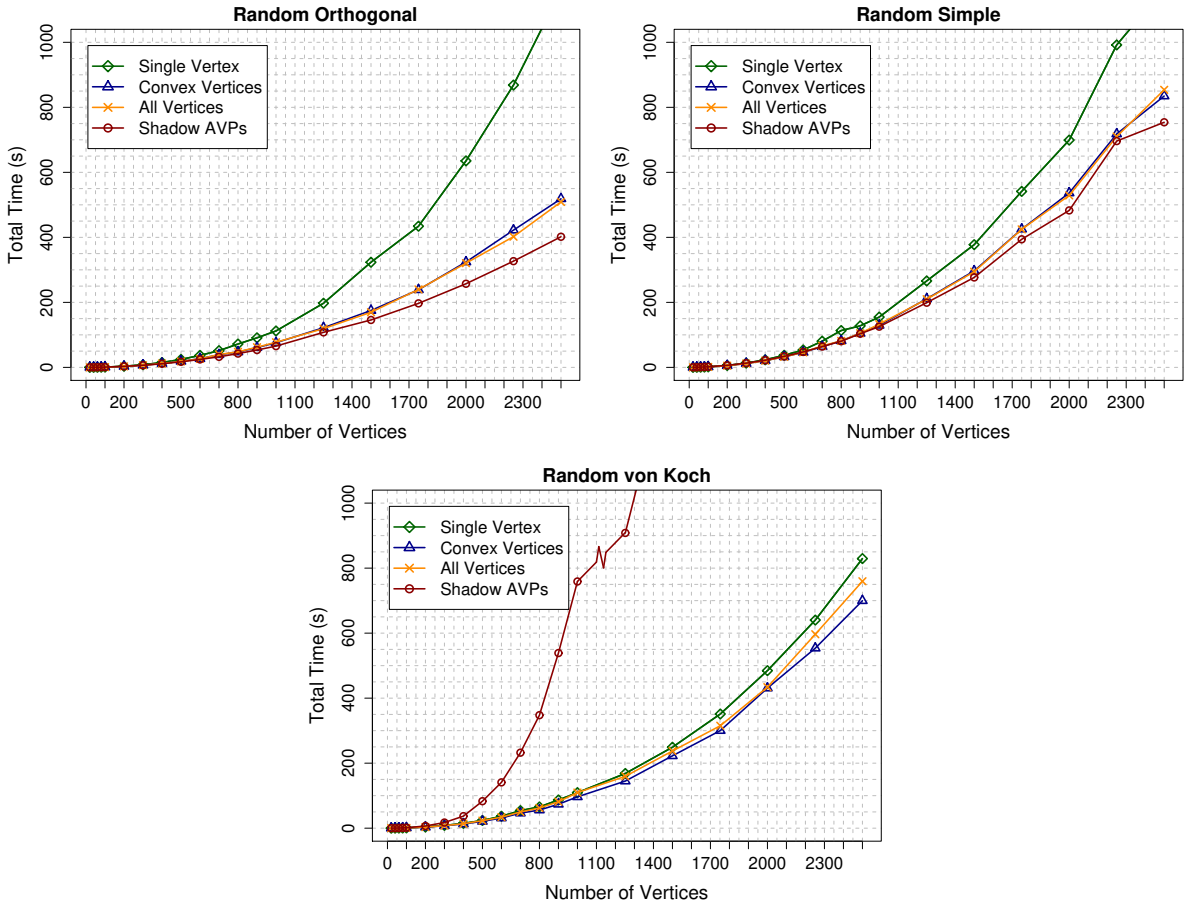


C4
(Pg.118)

Figure 6.12: A box-plot graph showing the number of iterations for Random von Koch polygons using the *Convex Vertices* strategy.

Figure 6.13 shows the total amount of time, including preprocessing and processing phases, to solve instances from the three random classes of polygons. Notice that all charts are plotted on the same scale. For a proper analysis of this chart, one has to bear in mind our previous discussion on the number of iterations and the size of the discretizations produced by each of the alternative strategies.

Firstly, notice that the *Convex Vertices* and *All Vertices* strategies lead to very similar computation times. Earlier, we had seen that their iteration counts are small and very close together. We also observed that the size of the final discretizations grows slowly (almost linearly) with the instance size and that the one corresponding to the *All Ver-*



C5
(Pg.118)

Figure 6.13: Total time by polygon type.

tices strategy is just a bit larger than the one for the *Convex Vertices* strategy. These similarities are due to the fact that, in both cases, the IP solver has to compute a lighter SCP instance at each iteration of the algorithm. The algorithms emerging from these two strategies are not only fast but also very robust. To see that, notice that the curves for *All Vertices* and *Convex Vertices* strategies remain absolutely similar as the instance classes vary.

On the other hand, the *Single Vertex* strategy behaves poorly for the *Random Orthogonal* and *Random Simple* classes. Though it always yields the smallest discretizations on average, the number of iterations required by this strategy grows very rapidly. Even though one might also expect light SCP instances to be optimized at each iteration, the overhead of multiple calls to the IP solver surpasses the benefit of small instances. Only for the RvK polygons the *Single Vertex* strategy becomes competitive with the *Convex Vertices* and *All Vertices* ones which is predictable since, for these polygons, we see that

the *Single Vertex* strategy usually executes just 10 iterations over the average of the two other strategies.

Now, we analyze the behavior of the algorithm under the *Shadow AVPs* strategy. This variant of the algorithm outperforms all the others for the *Random Orthogonal* and *Random Simple* classes, but it becomes excessively slow for RvK instances with just a few hundred vertices. To explain these results, we refer again to Figure 6.10.

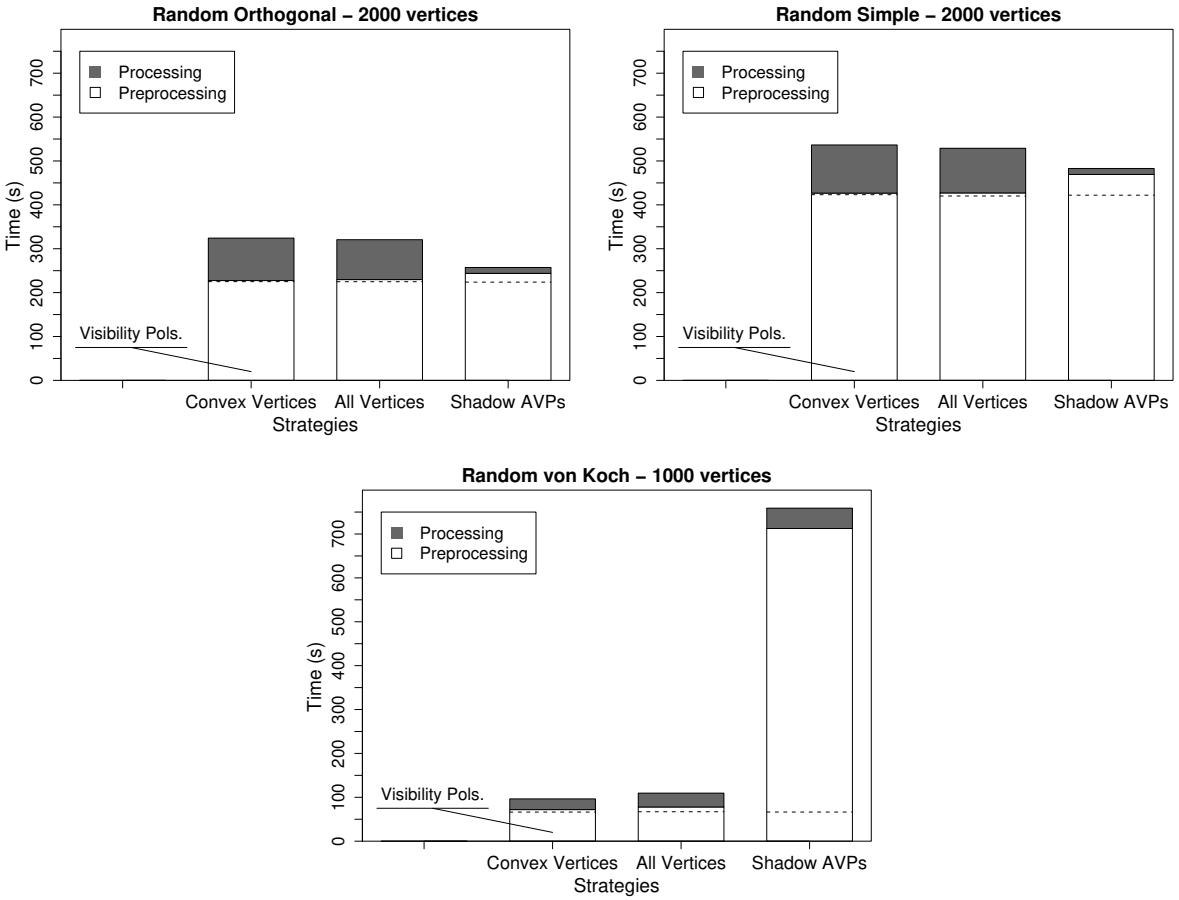
In the RvK class, recall that the number of shadow AVPs grows rapidly with the instance size. Thus, although a single SCP instance is solved, the time spent in the computation of the constraint matrix associated to that instance is enormous. Let us briefly defer the discussion of this last issue.

In Figure 6.10 we have seen that, for the *Random Orthogonal* and *Random Simple* classes, the final discretization or, similarly, the number of shadow AVPs grows almost linearly with the instance size. Actually, it is not much larger than the size of the final discretizations yielded by the *Convex Vertices* and *All Vertices* strategies. However, we know that a single iteration of the algorithm is enough in this case. Therefore, the size of the unique SCP instance to be solved is not much larger than that of the one solved in the last iteration of the *All Vertices* and *Convex Vertices* strategies.

We now turn our attention to the time spent by the algorithm in each phase relative to the discretization strategies. Recall that the preprocessing phase is composed of three procedures. The first one is common to all strategies and computes the visibility polygons of each vertex. The second one computes the initial discretization and its cost is highly affected by the choice of the strategy to be implemented. The worst case corresponds to the *Shadow AVPs* strategy since it requires the computation of all AVPs and the determination of the shadow ones along with their centroids. On the other extreme, we have the *Single Vertex* and *All Vertices* strategies where no computation is needed for the second procedure while, for the *Convex Vertices* strategy, some inexpensive calculations are required to determine which vertices are convex. Finally, in the third procedure of the preprocessing phase one has to build the starting IP model and the time spent in doing so depends on the size of the discretization. This clearly benefits the *Single Vertex* strategy and also, though to a minor extent, the *Convex Vertices* and *All Vertices* strategies.

Figure 6.14 details the computation times of *Random Orthogonal* and *Random Simple* polygons on 2000 vertices and RvK polygons on 1000 vertices. Notice that the same scale is used on the three charts to facilitate comparisons, this being the reason why smaller instances in the RvK class were considered. The bars in these charts highlight the fraction of the total time spent on the processing and preprocessing phases and, for the latter, the fraction consumed by the procedure that computes visibility polygons.

One can see that the time spent in the preprocessing phase is in accordance with the discussion above, the *Shadow AVPs* strategy being the most time consuming for RvK



C6
(Pg.118)

Figure 6.14: Break up of the execution time into processing and preprocessing, for polygons of the random classes.

polygons. For the *Random Orthogonal* and *Random Simple* classes, the preprocessing time for the *Shadow AVPs* strategy is about the same as those for the other two strategies and its advantage only shows in the processing phase. The first two charts are also illustrative of the fact that random simple polygons have more complex visibility structure than those in the orthogonal class.

What is somehow surprising is that, although we are solving NP-hard problems in the solution phase, in all cases the majority of the time expenditure takes place in the preprocessing phase, which is entirely polynomial. The extraordinary developments of IP solvers together with the fact the SCP instances arising from the AGP are among the easier ones explains this seeming counterintuitive behavior of the algorithm. Thus, a breakthrough in the performance of our algorithm would be attained if one could devise a discretization obtainable through a very fast procedure and, at the same time, satisfying

the property that a single iterations is enough to reach the optimum of an AGP instance. Comparing the sizes of the final discretizations of the different strategies shown earlier, there seems to be room for such improvements.

6.6 Conclusions and Remarks

In this paper, we compiled and extended our research on exact approaches for solving the Art Gallery problem (AGP). In prior works [1, 2], we focused on galleries represented by orthogonal polygons and proposed an algorithm, also discussed in Section 6.3, based on successive discretizations of the input polygon. Our earlier computational experiments were constrained to instances of no more than a thousand vertices, while here, we extended the algorithm to handle non orthogonal polygons and tested it with instances of up to 2500 vertices. Moreover, we proposed new discretization strategies since the algorithm is very sensitive to the choice of discretizations. As a result, we introduced the *Convex Vertices* strategy which presents the best performance seen so far.

We recall that the exact algorithm relies not only on the discretization of the interior of the input polygon, but also on the modeling of this simplified discrete problem as a Set Cover problem (SCP). The resulting SCP instance is solved to optimality by an IP solver and, if uncovered regions remain, additional constraints are included and the process is repeated. Clearly, the performance of the algorithm depends also on the number of such iterations.

While focusing on novel strategies to implement the discretization step, a thorough experimentation was carried out to assess the trade-off between the number of iterations and the time spent by the many variants of the algorithm that arise from the alternative discretization methods.

Confirming the results from our earlier works, the proposed algorithm had excellent overall performance. It also proved to be robust, in the sense that it was able to tackle instances from a broad range of polygon classes. Moreover, the fastest variants of the algorithm very quickly found solutions to instances of more than 2000 vertices. This more than doubled the size of the largest instances we had previously solved which, in turn, were five times larger than those reported earlier in the literature.

The *Convex Vertices* strategy proposed in Section 6.4 yields sparse discretizations and, as a consequence, small SCP instances. As it can be seen in Table 6.2, this leads to a very fast implementation for instances of up to 2500 vertices.

On the other hand, as we observed in [2], the apparent advantage of a discretization which ensures an exact solution after a single iteration of the algorithm has not been verified. In particular, this occurred with the *Shadow AVPs* strategy whose inefficiency was due to the expensive preprocessing phase in which the shadow AVPs are computed. For

Table 6.2: Total Time (in seconds) for the *Convex Vertices* strategy.

	Polygons Classes			
n	Random Ortho	Random Simple	RvK	CvK
100	0.74	1.38	0.76	0.84
500	18.64	32.48	20.82	21.09
2500	518.60	834.78	699.74	720.99

these computations, we employed a polynomial time algorithm implemented with powerful data structures and efficient library packages for performing the necessary geometric operations. And yet, we could not significantly lower the preprocessing time of the exact algorithm under the *Shadow AVPs* strategy.

Contrary to what was expected, in the case of *Shadow AVPs*, preprocessing remained more costly in time than the solution of the SCP instance, a well-known NP-hard problem. One could credit the extraordinary developments of IP solvers in recent years with the success of this algorithm. The advances in this field made possible the solution of large instances of SCP in very small amounts of time.

It remains an open question whether we can find yet another discretization leading to a single iteration of the algorithm, which is computable in time bounded by a very small degree polynomial on the number of vertices. This is a promising topic for future research which might be beneficial to our algorithm.

As a closing remark, we point out that we focused here on the vertex-guard problem since this is probably the variant of the general art gallery problem which has received the most attention in the literature to date, although an exact algorithm had still been lacking. Nevertheless, the methods presented here can certainly be adapted to any instance in which the guard candidates form a feasible finite set. Furthermore, the relevant case of simple polygons with holes which were not treated in this paper requires a more sophisticated implementation for an efficient computation of visibility polygons. This is one of the directions of our continuing research on this topic.

References

[1] M. C. Couto, C. C. de Souza, and P. J. de Rezende. An exact and efficient algorithm for the orthogonal art gallery problem. In *Proc. of the XX Brazilian Symp. on Comp. Graphics and Image Processing*, pages 87–94. IEEE Computer Society, 2007.

[2] M. C. Couto, C. C. de Souza, and P. J. de Rezende. Experimental evaluation of an exact algorithm for the orthogonal art gallery problem. In *WEA*, volume 5038 of *Lecture Notes in Computer Science*, pages 101–113. Springer, 2008.

[3] M. C. Couto, P. J. de Rezende, and C. C. de Souza. An IP solution to the art gallery problem. In *Proc. of the 25th Annual ACM Symposium on Computational Geometry*, 2009.

[4] M. C. Couto, P. J. de Rezende, and C. C. de Souza. An IP solution to the art gallery problem. Video published in the *18th Annual Video/Multimedia Review of Computational Geometry*, as part of the *25th Annual ACM Symposium on Computational Geometry*, 2009.

[5] R. Honsberger. *Mathematical Gems II.* Number 2 in The Dolciani Mathematical Expositions. Mathematical Association of America, 1976.

[6] V. Chvátal. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B*, 18:39–41, 1975.

[7] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[8] T. C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[9] J. Urrutia. Art gallery and illumination problems. In Jörg-Rüdiger Sack and Jorge Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. North-Holland, 2000.

[10] S. K. Ghosh. *Visibility Algorithms in the Plane.* Cambridge University Press, New York, 2007.

[11] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Trans. Inf. Theor.*, 32(2):276–282, 1986.

[12] J. Kahn, M. M. Klawe, and D. Kleitman. Traditional galleries require fewer watchmen. *SIAM J. Algebraic Discrete Methods*, 4:194–206, 1983.

[13] D. Schuchardt and H.-D. Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Mathematical Logic Quarterly*, 41:261–267, 1995.

[14] J.-R. Sack and G. T. Toussaint. Guard placement in rectilinear polygons. In G. T. Toussaint, editor, *Computational Morphology*, pages 153–175. North-Holland, 1988.

[15] H. Edelsbrunner, J. O'Rourke, and E. Welzl. Stationing guards in rectilinear art galleries. *Comput. Vision Graph. Image Process.*, 27:167–176, 1984.

[16] S. K. Ghosh. Approximation algorithms for art gallery problems. In *Proc. Canadian Inform. Process. Soc. Congress*, 1987.

[17] S. K. Ghosh. Approximation algorithms for art gallery problems in polygons. *Discrete Applied Mathematics*, 158(6):718–722, 2010.

[18] S. Eidenbenz. Approximation algorithms for terrain guarding. *Inf. Process. Lett.*, 82(2):99–105, 2002.

[19] Y. Amit, J. S. B. Mitchell, and E. Packer. Locating guards for visibility coverage of polygons. In *Proc. Workshop on Algorithm Eng. and Experiments*, pages 1–15, 2007.

[20] U. M. Erdem and S. Sclaroff. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Comput. Vis. Image Underst.*, 103(3):156–169, 2006.

[21] A. P. Tomás, A. L. Bajuelos, and F. Marques. On visibility problems in the plane - solving minimum vertex guard problems by successive approximations. In *Proc. of the 9th Int. Symp. on Artificial Intelligence and Mathematics*, 2006.

[22] M. C. Couto, C. C. de Souza, and P. J. de Rezende. Instances for the Art Gallery Problem, 2009. *www.ic.unicamp.br/∼cid/Problem-instances/Art-Gallery*.

[23] D. S. Johnson. A theoretician's guide to the experimental analysis of algorithms. In M. H. Goldwasser et al., editor, *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implem. Challenges*, pages 215–250. AMS, Providence, 2002.

[24] C. C. McGeoch and B. M. E. Moret. How to present a paper on experimental work with algorithms. *SIGACT News*, 30, 1999.

[25] P. Sanders. *Presenting data from experiments in algorithmics*, pages 181–196. Springer-Verlag New York, Inc., New York, NY, USA, 2002.

[26] B. Moret. Towards a discipline of experimental algorithmics. In *Proc. 5th DIMACS Challenge*.

[27] D. T. Lee. Visibility of a simple polygon. *Comput. Vision, Graphics, and Image Process.*, 22:207–221, 1983.

[28] B. Joe and R. B. Simpson. Visibility of a simple polygon from a point. Report CS-85-38, Dept. Math. Comput. Sci., Drexel Univ., Philadelphia, PA, 1985.

[29] B. Joe and R. B. Simpson. Correction to Lee's visibility polygon algorithm. *BIT*, 27:458–473, 1987.

[30] L. A. Wolsey. *Integer Programming*. Wiley-Interscience, 1998.

[31] T. Baumgartner, S. P. Fekete, A. Kröller and C. Schmidt. Exact Solutions and Bounds for General Art Gallery Problems. In *Proc. 12th Workshop on Algorithm Engineering and Experiments (ALENEX)*. 2010

[32] P. Bose, A. Lubiw, and J. I. Munro. Efficient visibility queries in simple polygons. *Computational Geometry*, 23(3):313–335, 2002.

[33] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[34] Xpress–optimizer. www.dashoptimization.com.

[35] K. Falconer. *Fractal Geometry, Mathematical Foundations and Applications*. John Wiley & Sons, 1990. pp. 120–121.

[36] A. P. Tomás and A. L. Bajuelos. Generating random orthogonal polygons. In *Current Topics in Artificial Intelligence*, volume 3040 of *LNCS*, pages 364–373. Springer, 2004.

Comentários

C1: Instâncias Complexas de Galerias Reais

Com inúmeras construções complexas existentes, é necessária uma estratégia de discretização inicial que permita a solução de instâncias de maior complexidade ainda em tempo aceitável. Observando-se, por exemplo, a Catedral de St. Sernin, na França, tem-se em sua planta baixa uma galeria complexa de 232 vértices, com um arranjo de visibilidade de 51788 AVPs, o que faz com que a instância do SCP gerada fique enorme. Entretanto, apenas 1303 destes são AVPs de sombra e, portanto, necessários para garantir que o algoritmo convirja em apenas uma iteração, tornando essa estratégia de discretização mais adequada ao caso. Verificou-se que são necessários 12 guardas para cobrir a planta baixa da Catedral de St. Sernin e que, ao utilizar-se uma estratégia mais leve, que requeresse mais iterações, obteve-se que, além do problema ser resolvido em apenas 12 iterações, o tamanho final da discretização foi de somente 182 pontos. Comparando esses resultados, percebe-se que ainda há um grande caminho a ser percorrido na busca por novas estratégias.

C2: Outras Estratégias

As estratégias descritas nesse trabalho como obsoletas podem ser vistas em detalhes no Capítulo 3, Seção 3.4 e também nos trabalhos anteriores. Essas estratégias são as de Grade Regular e de Grade Induzida e fizeram parte da pesquisa inicial. A primeira utilizava a densidade de pontos dentro do polígono para evitar espaços difíceis de serem cobertos, enquanto a segunda buscava encontrar esses espaços utilizando uma menor quantidade de pontos para discretizar o polígono, mas baseando-se em uma propriedade geométrica do mesmo.

C3: CvK e RvK de 2500 vértices $\times$ AVPs de sombra

Com o aumento do número de vértices das instâncias complexas, o arranjo de visibilidade necessário para determinar os AVPs de sombra cresce rapidamente em tamanho. Esse fato, aliado com o de produzir e carregar o modelo PLI da instância gerada, fez com que a memória limite para a execução dos testes dos polígonos completos e aleatórios de von Koch de 2500 vértices fosse atingida e, para evitar poluir os resultados com o tempo gasto em *swap*, essas instâncias não foram incluídas. Para contornar esse problema, sugere-se como continuidade deste trabalho uma otimização no gerenciamento de memória da implementação feita do algoritmo.

C4: *Outliers* em Gráfico de Números de Iterações

Observando a formação dos polígonos aleatórios de von Koch, percebe-se que a escolha de determinada instância se faz dentro de um universo muito grande de opções, em especial para os níveis maiores, posto que a quantidade de arestas no polígono completo obedece a equação $n = 4 \times 5^l$, onde l é o nível desejado. Isto faz com que o grande aumento do número de arestas entre um nível e o próximo acarrete em grandes discrepâncias dentro de um grupo de instâncias aleatórias. Ao juntar este fato com a impossibilidade de se resolver determinadas instâncias de 2500 vértices devido a insuficiência de memória, tem-se a justificativa para a grande quantidade de *outliers* no gráfico apresentado.

C5: Descontinuidade no Gráfico de Tempo Total para Instâncias RvK

A descontinuidade apresentada no gráfico é justificada pela diferença na geração das instâncias RvK para valores de n maior que 1000 (ver Capítulo 2, Seção 2.4), fazendo com que as instâncias maiores sejam mais difíceis de se resolver, graças à adição de mais um nível ao polígono, o que aumenta a complexidade do arranjo de visibilidade.

C6: Composição do Tempo de Execução do Algoritmo

Para mais detalhes, ver Comentário 4 na página 87.

Capítulo 7

Considerações Finais

Nesta dissertação, fizemos um amplo estudo de um Problema de Galeria de Arte com o objetivo de desenvolver um algoritmo exato capaz de resolvê-lo. Dentre as muitas variantes desse problema, o foco foi dado àquela onde é desejada a minimização do número de guardas estacionários em vértices do polígono que representa a galeria. Esse polígono pode ser do tipo ortogonal (OAGP) ou simples (AGP) sem obstáculos. Note-se que esse problema é NP-DIFÍCIL, o que torna ainda mais desafiador o projeto de um algoritmo exato que possa ser usado na prática.

O trabalho apresentado engloba diversos aspectos de um estudo aprofundado do problema. Começando com uma revisão dos conceitos geométricos e combinatórios necessários ao bom entendimento da pesquisa que foi realizada, o texto provê uma descrição formal do AGP, a qual é acompanhada de uma ampla revisão bibliográfica. Em seguida, descreve passo-a-passo o algoritmo que foi proposto, fornecendo as provas de sua exatidão e de convergência para uma solução ótima. Diversos aspectos práticos da implementação são discutidos em detalhes e uma avaliação experimental do algoritmo é reportada, acompanhada de uma análise meticulosa dos resultados alcançados. Ao final, foi demonstrada a viabilidade de utilização do método proposto, que mostrou-se capaz de computar à otimalidade instâncias relativas a galerias de arte com milhares de vértices.

7.1 Resumo do trabalho realizado

Nesse trabalho apresentou-se um algoritmo exato para a resolução de um Problema de Galeria de Arte. Além disso, duas reduções do AGP para o Problema de Cobertura de Conjuntos – uma redução de Karp e outra de Turing – são mostradas e fornecem a fundamentação teórica que assegura a corretude do algoritmo proposto.

Uma extensa experimentação de uma implementação desse algoritmo, com o intuito de validar seu uso prático, está descrita nos diversos artigos apresentados em conferências

internacionais e também reproduzidos no texto.

Inúmeras instâncias de sete diferentes tipos de polígonos com até 2500 vértices – mais de 10 vezes o tamanho das maiores instâncias resolvidas exatamente na literatura até então – foram utilizadas nos testes computacionais realizados com o algoritmo. Todas elas podem ser encontradas e baixadas de um portal *online* – *www.ic.unicamp.br/∼cid/Problem-instances/Art-Gallery*. Até onde sabemos, esse portal é único do gênero e, além das instâncias, exibe também as soluções geradas pelo nosso algoritmo.

Por fim, foram analisadas nessa dissertação diversas estratégias para a construção da discretização inicial do polígono que representa a galeria de modo a fazer com que o algoritmo convirja no menor tempo possível para uma solução ótima.

Como resultado, chegou-se a um algoritmo capaz de resolver de forma rápida e exata instâncias do problema com milhares de vértices. Utilizando-se a estratégia de discretização com o melhor comportamento geral (*Convex Vertices*), isso é conseguido em um tempo de computação pequeno e com uma quantidade de iterações muito baixa – ordens de grandeza aquém do limite teórico de pior caso.

7.2 Principais contribuições e publicações

Esta dissertação apresenta diversas contribuições diretas e indiretas para a resolução do Problema de Galeria de Arte. Entre elas, destacam-se:

- A implementação de um algoritmo de visibilidade que trabalha com aritmética exata e lida corretamente com casos degenerados;
- A disponibilização da única biblioteca de instâncias de galeria de arte *online* conhecida, contendo todas as instâncias utilizadas e os resultados obtidos;
- Uma análise de diversas estratégias para a discretização inicial da galeria, e o impacto de cada uma na resolução do problema;
- Duas reduções – uma de Karp e outra de Turing – do Problema de Galeria de Arte para o Problema de Cobertura de Conjuntos;
- Um algoritmo que computa uma solução ótima para o problema de cobertura mínima da galeria por guardas estacionários em vértices;
- A identificação de instâncias derivadas da curva de von Koch modificada, que mostraram-se bem mais difíceis de serem resolvidas que as existentes na literatura;
- Um estudo experimental completo sobre a implementação do algoritmo proposto e sua viabilidade prática;
- A resolução de instâncias complexas, com mais de 10 vezes o número de vértices daquelas encontradas na literatura, em menos de 800 segundos;
- A identificação dos desafios de implementação de um algoritmo robusto e exato para o problema proposto.

A qualidade dessas contribuições pode ser mensurada pelas publicações decorrentes do trabalho e listadas a seguir:

1. M. C. Couto, P. J. de Rezende, and C. C. de Souza, *An exact algorithm for an art gallery problem.* Technical Report IC-09-46, **Institute of Computing, University of Campinas**, November 2009. Em inglês, 25 páginas.
2. M. C. Couto, P. J. de Rezende, and C. C. de Souza. *An ip solution to the art gallery problem.* In **SCG '09: Proceedings of the 25th annual symposium on Computational geometry**, páginas 88–89, New York, NY, USA, 2009. ACM.
3. M. C. Couto, C. C. de Souza, and P. J. de Rezende. *Strategies for optimal placement of surveillance cameras in art galleries.* In **GraphiCon 2008: XI International Conference on Computer Graphics & Vision**, volume 1, page `http://www.graphicon.ru/2008/proceedings/technical.html`. Lomonosov Moscow State University, 2008.
4. M. C. Couto, C. C. de Souza, and P. J. de Rezende. *Experimental evaluation of an exact algorithm for the orthogonal art gallery problem.* In **WEA**, **Lecture Notes in Computer Science**, volume 5038, páginas 101–113. Springer, 2008.
5. M. C. Couto, C. C. de Souza, and P. J. de Rezende. *An exact and efficient algorithm for the orthogonal art gallery problem.* In **Proc. of the XX Brazilian Symp. on Comp. Graphics and Image Processing**, páginas 87–94. IEEE Computer Society, 2007.

7.3 Trabalhos Futuros

Embora o trabalho apresentado tenha sido extenso, existem várias possibilidades de continuação para essa pesquisa. Por exemplo, pode-se adaptar o algoritmo para que ele passe a aceitar instâncias compostas por polígonos contendo obstáculos no seu interior. Essa adaptação deve ocorrer, em especial, na rotina que gera as regiões de visibilidade, pois, uma vez que essas estejam corretamente calculadas, o restante do algoritmo está apto a resolver o problema sem sofrer alterações. A Figura 7.1 mostra a interface visual com o correto tratamento de uma instância de polígono simples com obstáculos. Repare que os guardas de uma das soluções ótimas estão destacados, além da região de visibilidade de um dos pontos internos da instância.

Uma outra evolução natural do trabalho é a inclusão de paralelismo na implementação do código. Isso pode ser obtido alterando uma dentre as muitas partes do código, em especial aquela responsável pela construção das regiões de visibilidade e aquela que monta a matriz a ser passada para o Problema de Cobertura de Conjuntos.

Na verdade, várias outras extensões interessantes desse trabalho poderiam ser elencadas. Encerramos esta dissertação sugerindo algumas delas:

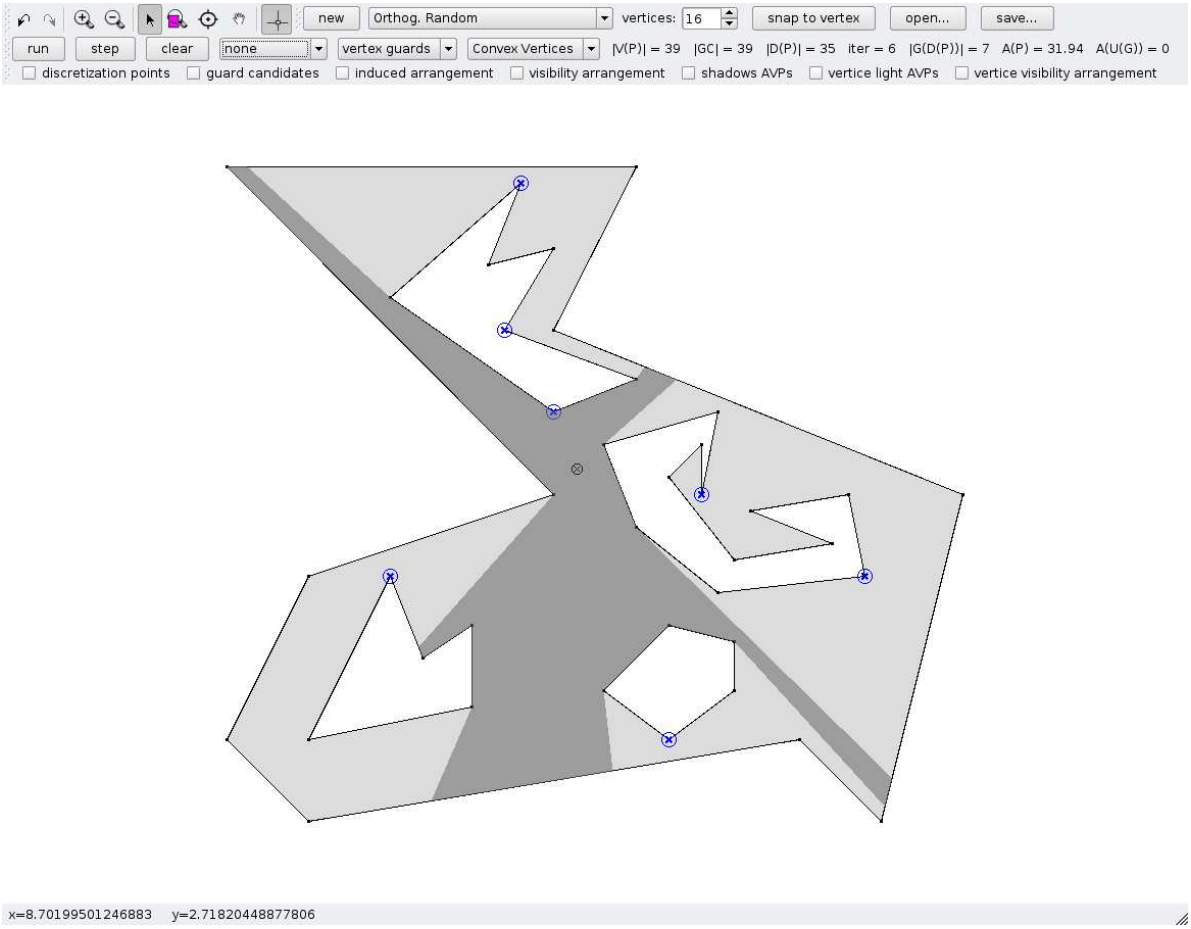


Figura 7.1: Trabalhos Futuros – Instância com Obstáculos.

- Identificação e eliminação de AVPs redundantes.
- Investigação de novas estratégias para a discretização inicial da galeria.
- Construção de instâncias possivelmente difíceis do Problema de Galeria de Arte baseadas na redução de cláusulas do 3-SAT [34].
- Utilização de um *software* livre que disponibilize um resolvedor de Programação Linear Inteira para otimizar o Problema de Cobertura de Conjuntos.
- Investigação de algoritmos para eliminação de vértices candidatos a guardas.
- Cobertura dupla dos vértices da galeria de arte, isto garante que todo guarda seja visto por algum outro guarda, gerando uma redundância com propriedades interessantes.
- Estudo da variante tridimensional do problema.
- Guardas Livres (*Point Guards*) para cobertura de vértices e para cobertura da galeria como um todo.

Appendix A

Strategies for Optimal Placement of Surveillance Cameras in Art Galleries

Prólogo

O artigo incluído neste apêndice foi escrito em co-autoria com os professores Cid C. de Souza e Pedro J. de Rezende, ambos do Instituto de Computação da Universidade Estadual de Campinas. Apresentado em Moscou, em outubro de 2009, durante o XIX GraphiCon (*International Conference on Computer Graphics and Vision*) e publicado nos anais daquela conferência, este artigo reflete um trabalho intermediário, entre os artigos correspondentes aos Capítulos 5 e 6, ou seja, reflete um passo antes da fase final da pesquisa, onde já trabalhava-se com polígonos simples, mas em uma escala muito reduzida e com instâncias do mesmo tamanho do trabalho anterior.

Abstract

The Art Gallery problem (AGP) consists of minimizing the number of cameras required to guard an art gallery whose boundary is an n-vertex polygon P. In this paper, we report our ongoing work in exploring an exact algorithm for a few variants of AGP, which iteratively computes optimal solutions to Set Cover problems (SCPs) corresponding to discretizations of P. Besides having proven in [6] that this procedure always converges to an exact solution of the original continuous problem, we have evidence that, in practice, convergence is achieved after only a few iterations, even for random polygons of hundreds of vertices. Nonetheless, we observe that the number of iterations required is highly dependent on the way P is initially discretized. As each iteration involves the solution of an SCP, the strategy for discretizing P is of paramount importance. We present here some of the discretization strategies we have been working with and new ones that will be studied in the near future. In comparison to the current literature, our results show a significant improvement in the size of the instances that can be solved to optimality while maintaining low execution times: no more than 65 seconds for random polygons of up to one thousand vertices.

A.1 Introduction

In 1973, Victor Klee posed the *Art Gallery Problem*, (AGP), which consists in determining the minimum number of cameras sufficient to guard the interior of an n-wall art gallery [10]. Chvátal showed, that $\lfloor n/3 \rfloor$ cameras are occasionally necessary and always sufficient to guard a simple polygon with n vertices [5].

We focus here on the specific variation of AGP in which the placement of cameras is restricted to the vertices of the polygon and they have a 360° field of vision. Since the corresponding minimization problem has been proven NP-hard [11] even for orthogonal polygons [14], placement of sub-optimal numbers of cameras has been studied [3, 13], as well as a few approximation algorithms [1]. Moreover, modeling AGP as a discrete combinatorial problem followed by the solution of the corresponding optimization problem has been attempted by Erdem and Sclaroff [9] who discretize the interior of the polygon using a fixed grid, yielding an approximation algorithm.

The approach that we have been studying consists of iteratively modeling the problem as a classical *Set Cover problem* (SCP). Among the results obtained so far, we have shown that the number of iterations executed by our method is polynomially bounded, and in practice, very small. Furthermore, the experimental analysis show that the number of iterations and the total time depends much on how the polygon is discretized. This becomes clearer when one realizes that at each iteration an instance of SCP, an NP-hard

problem, has to be solved to optimality, by an Integer Programming (IP) solver, in our case.

So far, we have completed a thorough experimental investigation concerning the trade-off between the number and nature of the discretizing points and the number of iterations, analyzing the practical viability of each approach. Our test data, available at [8], includes multiple instances for each size of the vertex set, for various classes of polygons with up to *a thousand* vertices. The experimental results obtained thus far surpass, by more than fivefold, those reported elsewhere in [16] and in [6].

In the next sections, we present the method, we summarize the experimental results, we describe our testing environment and we indicate venues of further investigation and a conclusion.

A.2 Algorithm

In an instance of AGP we are given a simple polygon P that bounds an art gallery and we are asked to determine the minimum number and an optimal placement of vertex cameras in order to keep the whole gallery under surveillance. Vertex cameras are assumed to have a range of vision of 360°.

The approach used by the algorithm described below transforms the continuous AGP into a discrete problem which, in turn, can be easily modeled as an instance of SCP.

Due to limited space, we will assume the reader's familiarity with the standard terminology from the literature, such as [12]. However, the following notation is best made explicit: a *vertex surveillance set* S is any subset of vertices of P such that $\bigcup_{s \in S} V(s) = P$, where $V(s)$, the *visibility region of* s, is the set of all points visible from vertex s. In other words, a vertex surveillance set for P gives the positions of stationary cameras which can oversee an entire art gallery of boundary P. Thus, AGP amounts to finding the smallest subset S of vertices that is a vertex surveillance set for P. compute the visibility polygons.

We now describe how the solutions to successively refined discrete instances of SCP are guaranteed to converge to an optimal solution to the original continuous problem. To this end, consider an arbitrary discretization of P into a finite set of points $D(P)$. An IP formulation of the corresponding SCP instance is:

$$\begin{aligned} z \;=\; \min \;& \sum_{j \in V} x_j \\ \text{s.t.} \;& \sum_{j \in V} a_{ij} x_j \geq 1, \text{ for all } p_i \in D(P) \qquad (1) \\ & x_j \in \{0,1\}, \text{ for all } j \in V \end{aligned}$$

where the binary variable x_j is set to 1 if and only if vertex j from P is chosen to be in the surveillance set. Moreover, given a point p_i in $D(P)$ and a vertex j of P, a_{ij} is a

binary value which is 1 if and only if $p_i \in V(j)$.

Given a feasible solution x for the IP above, let $Z(x) = \{j \in V \mid x_j = 1\}$. Constraint (1) states that each point $p_i \in D(P)$ is visible from at least one selected camera position in the solution and the objective function minimizes the cardinality z of $Z(x)$. Clearly, as the set $D(P)$ is finite, it may happen that $Z(x)$ does not form a vertex surveillance set for P. In this case, we must add a new point inside each uncovered region and include these points in $D(P)$. A new SCP instance is then created and the IP is solved again.

We are now able to describe the algorithm proposed in [6]: in the *preprocessing phase*, two procedures are executed. The first one computes the visibility polygons for the points in V. The second one computes the initial discretization $D(P)$ and builds the corresponding IP model. In the *solution phase*, the algorithm iterates as described above, solving SCP instances for the current discretization, until no regions remain uncovered.

We had shown in [6] that an upper bound on the number of iterations is $O(n^3)$ which derives from the fact that the edges of the visibility regions induce a subdivision of P which is comprised of $\Theta(n^3)$ faces or *Atomic Visibility Polygons* (AVPs) – see [4]. One point inside an AVP is enough to guarantee that this entire AVP will be covered by the solution to the discretized problem. Whence follows the upper bound on the number of iterations.

Moreover, the actual number of iterations that is required depends on how many uncovered regions can be successively generated. As the cost of each iteration is related to the number of constraints in (1), an interesting trade-off naturally sprouts and leads one to attempt multiple choices of discretization schemes. On the other hand, any method of cleverly choosing the initial points of the discretization will have a corresponding cost in preprocessing time, opening another intriguing time exchange consideration. These questions are precisely what we address next.

In Section A.3 we briefly describe several discretization schemes leading to the various performance analysis summarized in Section A.4.

A.3 Discretization Strategies

The key point in the IP approach is to set up instances of SCP that can rapidly be solved while minimizing the number of iterations required to attain an optimal solution to the original art gallery problem, within the least amount of time. However, one must take into account that sophisticated geometric properties used to build more efficient discretizations will generate a corresponding cost in preprocessing, possibly outweighing the benefits.

Regular Grid. The first discretization strategy considered here is based on the generation of a dense grid inside the polygon in the assumption that few iterations might

result. This approach, however, leads to very large instances of SCPs which increase the time needed to run the IP solver. A summary of the outcome of the use of regular grids for von Koch polygons can be seen in Table A.1.

Induced Grid. Given the perception that reflex vertices are partly responsible for the hardness of the problem, a natural discretization strategy is the grid induced by the edge extensions that intersect in the polygon. Here, we generate fewer constraints than in the previous strategy while capturing more of the intrinsic visibility information of the polygon. One can expect this to decrease the time to solve the instances.

Just Vertices. As an extreme case, consider the rather sparse grid consisting only of the vertices of P. Surprisingly, this strategy leads to an overall faster method than the two aforementioned ones, due to the fact that at each new iteration all added constraints correspond to areas harder to supervise and all SCP instances are small.

Reduced Atomic Visibility Polygons. It is easy to see that, by definition, if a camera surveillance set S covers the centroid of an AVP, then it must cover the entire AVP. Therefore, if S covers the centroids of every AVP of P, then S must be a surveillance set for P. As an initial discretization comprised of the centroids of all AVPs would lead to an impractically large instance of $O(n^3)$ constraints, we have devised a way to reduce it to an equivalent subset. We refer the reader to [7] for the details of this reduction.

A.4 Experimental Results

In this section, we summarize our experimental evaluation of the discretization strategies discussed above. A description of the testing environment is presented in section A.5.

A.4.1 Instances

We conducted the tests on a large number of instances, downloadable from [8], of polygons from three classes. The first two of these are composed of n-vertex orthogonal (random and von Koch) polygons and the last one is comprised of random simple (non-orthogonal) polygons.

(1) **Orthogonal Random:** These are n-vertex random ortho-polygons generated as described in [15].

(2) **Complete von Koch (CvK):** These polygons are based on a modified version of the fractal von Koch curve, which is generated, starting from a square, by recursively replacing each edge as shown in Figure A.2, where $\overline{ar} = \overline{st} = \overline{ub}$ and $\overline{sr} = \overline{tu} = \frac{3}{4}\overline{ar}$.

(3) **Simple Random:** This class consists of randomized simple (non orthogonal) polygons generated using [2]. The random instances were generated for the number of vertices

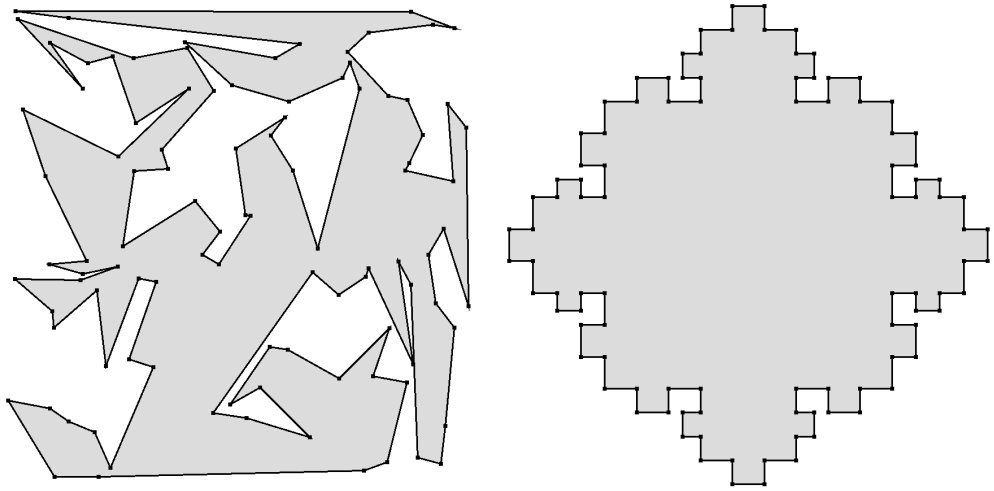


Figure A.1: Sample polygons: Simple Random and Orthogonal von Koch (with 100 vertices).

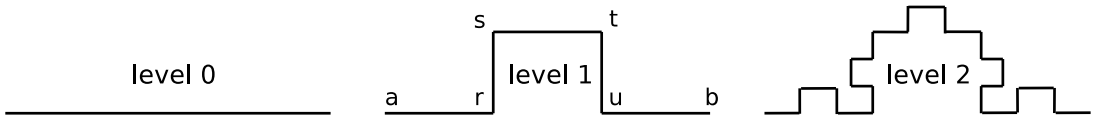


Figure A.2: Levels of modified von Koch polygons.

n in the ranges: [20, 200] with step 20, (200, 500] with step 50 and (500, 1000] with step 100. The CvK class contains, by construction, only 3 instances with $n \in \{20, 100, 500\}$.

To have statistical significance, we chose the number of instances generated to be between 10 and 30 for each value of n. Thus, in total, our data set is composed of 643 instances, having between 20 and 1000 vertices.

A.4.2 Results

We now discuss the actual experimental evaluation of the strategies described in Section A.3. All values reported here are average results for all instances of each size, or multiple runs of the same instance of CvK polygons.

The usage of discretization strategies based on dense grids becomes discouraging when we analyze the results in Table A.1 which displays the execution time and the size of the discretization of the strategies proposed in Section A.3 for the CvK polygons. One can see that for these instances, the *Induced Grid* strategy has a better performance than the *Regular Grid* strategy. The size of the discretization produced by *Regular Grid* grows quadratically in the number of vertices and inflates the number of constraints in the

Table A.1: Results for Complete von Koch polygons.

	Final $\|D(P)\|$			Total Time		
# vertices	20	100	500	20	100	500
Reg. Grid	45	500	6905	0.05s	1.57s	92.37s
Ind. Grid	24	205	1665	0.03s	1.41s	70.94s
Red. AVPs	28	324	5437	0.07s	3.14s	143.93s
Just Vert.	20	107	564	0.04s	0.97s	29.35s

IP formulation increasing considerably the time necessary to optimally solve the SCP instances. The *Reduced AVP* strategy has a poor behavior for CvK polygons since the number of shadow AVPs increases fast in this case. The *Just vertices* strategy is the one that spends less time.

Figure A.3 shows the amount of discretized points necessary for each strategy to achieve the optimal solution of AGP for random ortho-polygons. One can see that the *Regular Grid* strategy rapidly becomes impractical due to the huge size of the discretization and, therefore, will no longer be analyzed. On the other hand, the *Reduced AVP* strategy is very well-suited for random ortho-polygons.

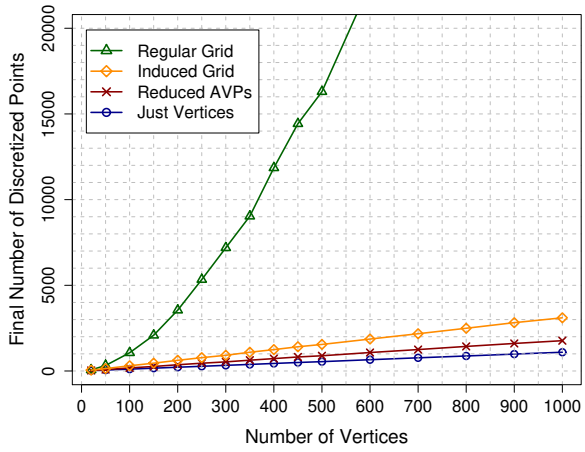


Figure A.3: Final discretization size: Random ortho-polygons.

The curves corresponding to the *Just Vertices* strategy suggest that the set of vertices of the polygon is a good guess for the initial discretization since few new points are added to it to achieve the optimal solution of an AGP instance for this class of instances.

Figure A.4 shows the number of iterations each strategy needs to achieve the optimal solution for random ortho-polygons. The chart displays the expected behavior with the number of iterations increasing as the size of the discretizations decrease. Now, relative to

the size of the input polygon, the number of iterations remains negligible when compared to the theoretical bound of $\Theta(n^3)$.

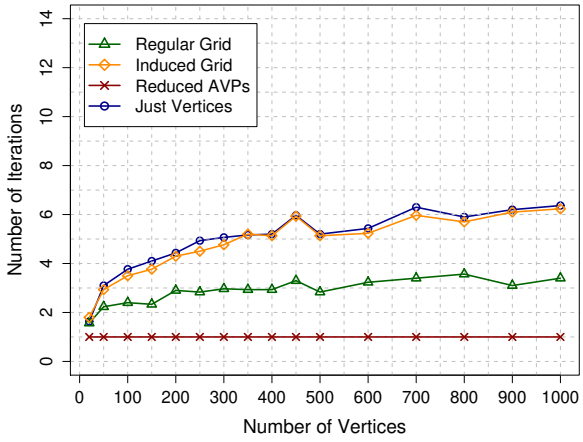


Figure A.4: Number of iterations: Random ortho-polygons.

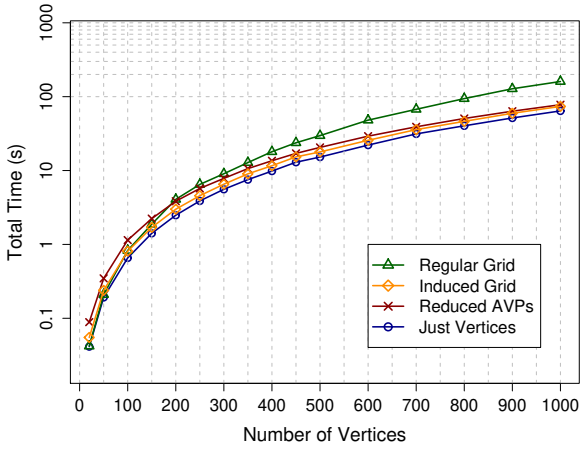


Figure A.5: Total time: Random ortho-polygons.

Figure A.5 shows the total time, including the preprocessing and processing phases, to solve instances from the random ortho-polygons. The curves are plotted in log × linear format and both charts are in the same scale. One can see that for Random polygons, all the strategies behave similarly except, the *Regular Grid*.

It is interesting to notice that the corresponding graph for simple random polygons in Figure A.6 is similar in shape, except that the times are doubled, due to this being a more difficult problem.

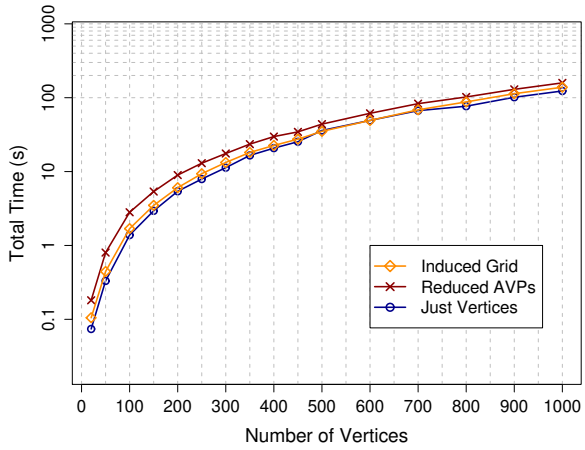


Figure A.6: Total time: Random simple polygons.

In Figure A.7 one can see that even though we are solving NP-hard problems (SCPs) in the solution phase, most of the computing time is spent in the preprocessing phase, which is polynomial both while constructing the visibility polygons and while assembling the IP model after setting up the initial discretization. Note that the *Just Vertices* strategy requires almost no computation after the visibility polygons are built.

A.5 Testing Environment

To evaluate our algorithm and the discretization strategies, we implemented, in `C++`, a testing environment designed in three layers, each one with a specific function. The first layer is a combination of the geometric library `CGAL 3.2.1`, the commercial IP solver `Xpress v17.01.02` and literature standard algorithms to construct visibility polygons and to generate random polygons. This layer also contains a strategy depot. The system's architecture is depicted in Figure A.8.

The second layer consists of the core of the application, i.e., the implementation of the method and uses several different interfaces for communication with the other layers and thus manages the algorithms that optimally solve the problem.

Finally, the third layer comprises the graphical interface of the application (see Figure A.9) containing several toolbars that allows the user to: change polygon type and the strategy used; run the method step by step with visual identification of the uncovered and covered regions; display the visibility region of any point inside the polygon; and zoom in to investigate details. It also displays information related to the discretization strategy and from the solution in view.

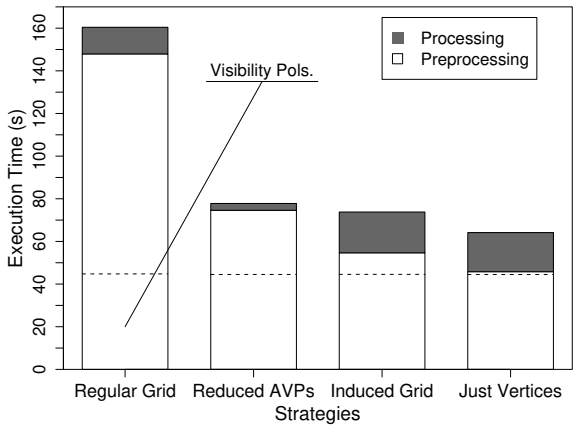


Figure A.7: Execution time for polygons of 1000 vertices: Random ortho-polygons.

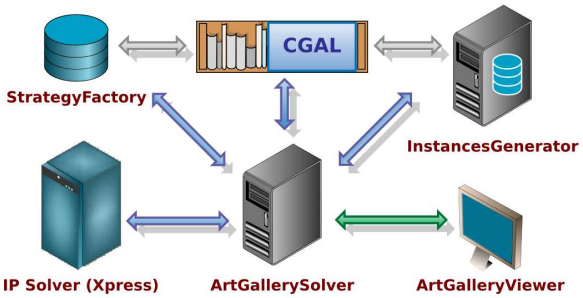


Figure A.8: Application Architecture.

As for hardware, we used a desktop PC featuring a Pentium IV at 3.4 GHz and 1 GB of RAM running `GNU/Linux 2.6.17`.

A.6 Conclusions and Remarks

We conducted an experimental investigation of an exact algorithm for the NP-hard Art Gallery problem which relies on the discretization of the interior of the input polygon P and on the modeling of this simplified discrete problem as a Set Cover problem (SCP). The resulting SCP instance is solved to optimality by an IP solver and, if uncovered regions remain, additional constraints are included and the process is repeated. Clearly, the performance of the algorithm depends on the number of such iterations.

This work focused on different strategies to implement the discretization of P. Thorough experimentation was carried out to assess the trade-off between the number of iterations and time spent by the many variants of the algorithm that arise from the alternative

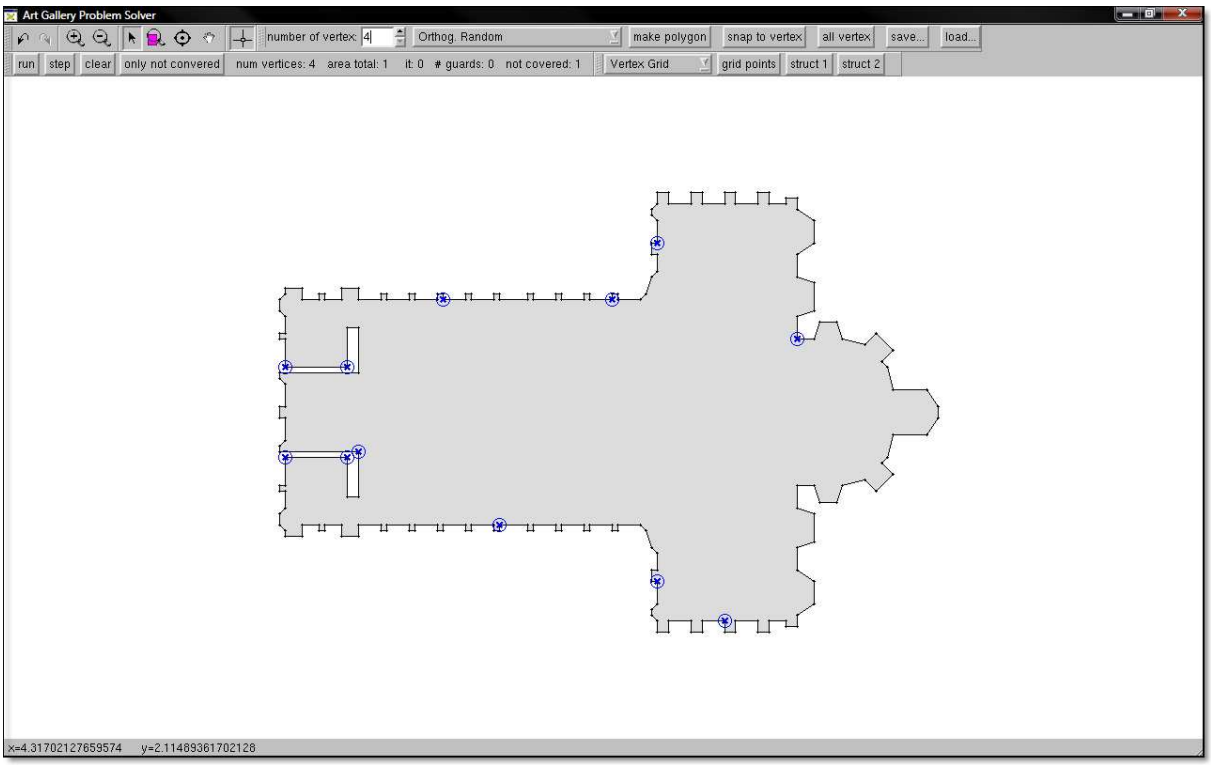


Figure A.9: Application Interface showing an optimal solution for a polygon of 232 vertices corresponding to a simplified floor plan of the Basilica of St. Saturnin in Toulouse, France.

discretization methods.

Our conclusion is that this exact algorithm is a viable choice to tackle instances of AGP, since the largest ones we solved were five times larger than those reported earlier in the literature.

Additional strategies for the initial discretization that are still under consideration include starting off with: *(i)* a single vertex, or a single internal point; *(ii)* only the reflex vertices, or only the convex vertices; *(iii)* only the centroids of AVPs that intersect edges of P.

Furthermore, our algorithm can solve (with minor modifications) a few related problems which we are currently working on:

- alternate sets of discrete potential camera spots (instead of just vertices), such as: *(i)* just reflex vertices, or just convex vertices; *(ii)* midpoints of all edges; *(iii)* a dense internal grid, leading to an approximate solution to the continuous placement problem.
- redundant coverage (e.g., double coverage of P).

On the other hand, a promising venue of further investigation lies in trying to identify inexpensive geometric properties that might lead to a set of constraints that capture the essence of the hardness of the problem, such as a significant reduction on the number of AVPs.

About the Authors

Marcelo C. Couto is a graduate student in Computer Science at UNICAMP. His e-mail is: couto.marcelo@gmail.com
Cid. C. de Souza is a professor at UNICAMP. His e-mail is: cid@ic.unicamp.br
Pedro J. de Rezende is a professor at UNICAMP. His e-mail is: rezende@ic.unicamp.br

Correspondence to the authors can be addressed to:
Instituto de Computação, C.P. 6176
Universidade Estadual de Campinas, UNICAMP
Campinas, SP, Brazil, 13083-970.

References

[1] Amit, Y., Mitchell, J. S. B., and Packer, E. 2007. Locating guards for visibility coverage of polygons. In *Proc. Workshop on Algorithm Eng. and Experiments*, 1–15.

[2] Auer, T., and Held, M., 1996. Heuristics for the generation of random polygons.

[3] Avis, D., and Toussaint, G. T. 1981. An efficient algorithm for decomposing a polygon into star-shaped polygons. *Pattern Recogn. 13*, 395–398.

[4] Bose, P., Lubiw, A., and Munro, J. I. 2002. Efficient visibility queries in simple polygons. *Comput. Geom. 23*, 3, 313–335.

[5] Chvátal, V. 1975. A combinatorial theorem in plane geometry. *Journal of Combinatorial Theory Series B 18*, 39–41.

[6] Couto, M. C., de Souza, C. C., and de Rezende, P. J. 2007. An exact and efficient algorithm for the orthogonal art gallery problem. In *Proc. of the XX Brazilian Symp. on Comp. Graphics and Image Processing*, IEEE Computer Society, 87–94.

[7] Couto, M. C., de Souza, C. C., and de Rezende, P. J. 2008. Experimental evaluation of an exact algorithm for the orthogonal art gallery problem. In *Proc. of the 7th International Workshop on Experimental Algorithms, WEA 2008*, Springer-Verlag, vol. 5038, 101–113.

[8] Couto, M. C., de Souza, C. C., and de Rezende, P. J., 2008. OAGPLIB - orthogonal art gallery problem library. `http://www.ic.unicamp.br/~cid/Problem-instances/Art-Gallery/`.

[9] Erdem, U. M., and Sclaroff, S. 2006. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Comput. Vis. Image Underst. 103*, 3, 156–169.

[10] Honsberger, R. 1976. *Mathematical Gems II.* No. 2 in The Dolciani Mathematical Expositions. Math. Assoc. of America.

[11] Lee, D. T., and Lin, A. K. 1986. Computational complexity of art gallery problems. *IEEE Trans. Inf. Theor. 32*, 2, 276–282.

[12] O'Rourke, J. 1987. *Art Gallery Theorems and Algorithms.* Oxford University Press.

[13] Sack, J.-R., and Toussaint, G. T. 1988. Guard placement in rectilinear polygons. In *Computational Morphology*, G. T. Toussaint, Ed. North-Holland, 153–175.

[14] Schuchardt, D., and Hecker, H.-D. 1995. Two NP-hard art-gallery problems for ortho-polygons. *Mathematical Logic Quarterly 41*, 261–267.

[15] Tomás, A. P., and Bajuelos, A. L. 2004. Generating random orthogonal polygons. In *Current Topics in Artificial Intelligence*, Springer, vol. 3040 of *LNCS*, 364–373.

[16] Tomás, A. P., Bajuelos, A. L., and Marques, F. 2006. On visibility problems in the plane - solving minimum vertex guard problems by successive approximations. In *Proc. of the 9th Int. Symp. on Artificial Intelligence and Mathematics.*

Apêndice B

Implementação do Algoritmo

Este Apêndice mostra a abordagem do Problema de Galeria de Arte do ponto de vista prático, utilizando como base a teoria do algoritmo proposto, descrito em detalhes no Capítulo 3. O processo de concepção da solução implementada é apresentado, incluindo o desenvolvimento das interfaces para a interação do usuário, que auxiliam a experimentação, avaliação e estudo mais apurado do algoritmo.

B.1 Arquitetura do Sistema

Dada a aplicação prática e a natureza geométrica do problema, é natural que se almejasse o desenvolvimento de códigos que simplificassem a experimentação, inclusive disponibilizando ferramentas de visualização de dados e de soluções. Foi com essa percepção que teve inicio a implementação do algoritmo proposto.

Da análise inicial, definiu-se também o esboço da arquitetura do sistema e gerou-se uma grande variedade de artefatos e documentos, necessários para garantir a qualidade, confiabilidade e adequação da implementação aos requisitos. As principais etapas e resultados da análise da arquitetura do sistema estão descritos a seguir.

B.1.1 Ambiente

Inicialmente, observou-se quais eram os requisitos de ambiente para a implementação e execução do aplicativo, obtendo-se um primeiro esboço da arquitetura do sistema.

Foram consideradas as diferenças entre os ambientes físicos disponíveis e os sistemas e bibliotecas existentes em cada um, bem como quais comportariam as outras bibliotecas necessárias. Ainda, foi analisado o quanto cada uma das configurações disponíveis impactaria positivamente a eficiência do algoritmo e a agilidade de implementação.

Por fim, também foi avaliada a facilidade da evolução e troca de ambiente, quando necessário. Como resultado, obteve-se o melhor ambiente disponível para desenvolver e testar experimentalmente a implementação do algoritmo proposto. Em cada trabalho publicado, esse ambiente está descrito em detalhes, o que facilita a reprodução dos resultados obtidos. A seguir, tem-se essa descrição do ambiente quando da escrita dessa dissertação.

Hardware.

Um dos mais importantes requisitos de *hardware* era que o sistema pudesse executar na maioria dos ambientes físicos disponíveis na atualidade. No entanto, não se deve fazer uso das vantagens que determinadas arquiteturas possam prover. Optou-se, portanto, por um PC desktop comum, com as seguintes configurações:

- Como processador, possuía um Intel® Pentium® D CPU 3.40 GHz (x86, *Type* 0, *Family* 15, *Model* 6, *Stepping* 4) que, embora disponibilize dois núcleos, teve apenas um deles utilizado pelo sistema.
- Quanto a memória, a disponibilidade era de 3 GB de RAM DDR2 a 667 MHz.

Nota-se, pela descrição do ambiente físico utilizado, que uma clara evolução do sistema é considerar o paralelismo de processadores para determinados cálculos. Com isso, pode-se ganhar um tempo considerável de execução (veja trabalhos [16, 11, 15]), principalmente no preprocessamento e, em especial, na determinação das regiões de visibilidade dos pontos candidatos a guarda.

Software.

Quanto aos requisitos de software, o principal era interfacear de maneira simples com resolvedores de programação linear e também com bibliotecas de algoritmos geométricos. Isso determinou que a linguagem de programação a ser utilizada fosse `C++`.

Ainda, vista a possibilidade do uso de um *framework* para compilação do código, o sistema tornou-se independente de sistema operacional, e pode ser utilizado em qualquer `SO` que suporte os softwares descritos ou similares – detalhados na Seção B.2. No caso dos resultados apresentados nessa dissertação, utilizou-se o `GNU/Linux 2.6.24` como sistema operacional e o `GNU g++ 4.2` como compilador.

B.1.2 Concepção

O algoritmo proposto nessa dissertação foi desenvolvido para o problema de minimização de guardas estacionários em vértices do polígono P que representa a galeria. Ele pode ser aplicado tanto para polígonos ortogonais sem obstáculos (OAGP), quanto para polígonos simples sem obstáculos (AGP).

O sistema desenvolvido deve avaliar experimentalmente esse algoritmo nas diferentes estratégias de discretização propostas. Ainda, ele deve ser capaz de mostrar informações

referentes à localização dos guardas da solução atual, quais regiões essa solução deixa descoberta, qual a estratégia em uso, entre outras informações relevantes que gerem evidências da eficiência do algoritmo e auxiliem a tomada de decisões.

Essas e outras informações, reportadas pelo sistema ao se encontrar uma solução ótima para uma instância, permitem que seja efetuada uma análise estatística do desempenho e da eficiência do algoritmo.

Da junção de todos os requisitos apresentados, foi desenvolvido um esboço da arquitetura do sistema, exibido na Figura B.1.

Perceba que a arquitetura foi desenvolvida para comportar facilmente evoluções do algoritmo, das estratégias de discretização e do ambiente. Isso deve-se ao uso de componentes específicos e responsáveis por cada função. Ainda, a componentização em camadas reduz a complexidade do design da arquitetura e aumenta a flexibilidade e a facilidade de manutenção do código.

B.1.3 Componentização em Camadas

Como mencionado, a componentização em camadas facilita a evolução, o projeto e os testes do sistema, pois separa as responsabilidades de cada componente e de cada camada, de forma que eles independam da implementação feita. Isso é possível devido à criação de interfaces para a comunicação entre os componentes, o que determina a atribuição de cada um.

O sistema desenvolvido pode ser particionado em três grandes camadas, cada qual responsável por um ponto específico do algoritmo. A primeira delas pode ser definida como um grande conjunto de bibliotecas que provêem suporte para a lógica do algoritmo.

Esta camada é composta por um resolvedor comercial de Programa Linear Inteiro, que pode ser facilmente substituído por um resolvedor *open-source*, e por uma biblioteca *open-source* de algoritmos geométricos. Além disso, contém alguns algoritmos – facilmente encontrados na literatura – e desenvolvidos especificamente para resolver pequenas tarefas, como gerar as instâncias aleatórias e calcular a região de visibilidade de um ponto qualquer. Por fim, existe uma *factory* que provê a implementação de cada uma das estratégias de discretização apresentadas.

A segunda camada contém o coração da aplicação. Nela, está a implementação do algoritmo proposto. Essa camada gerencia o uso dos objetos e das bibliotecas disponíveis, utilizando diferentes interfaces para essa comunicação. Com isso, é possível que o algoritmo seja executado como previsto e que a instância seja resolvida à otimalidade.

Por fim, tem-se a última camada, responsável pela visualização dos resultados e interação com o usuário. Foi implementada uma interface texto (linha de comando) e uma outra interface gráfica. Ambas estão descritas em detalhes na Seção B.5.

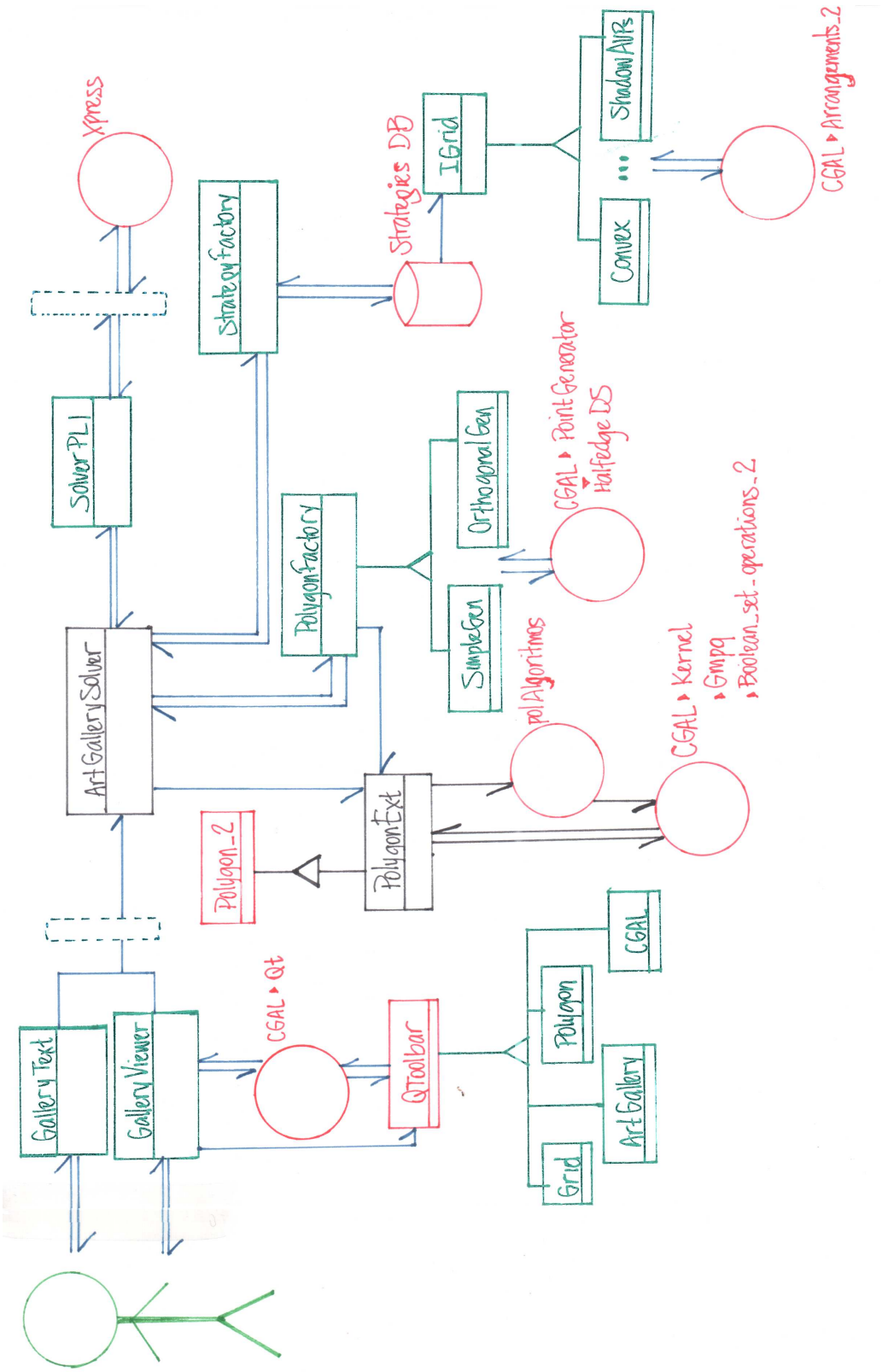


Figura B.1: Arquitetura da Aplicação

B.1.4 Componentes

Após a separação em camadas, é necessário definir e detalhar as atribuições de cada componente e a forma como será feita a interação entre eles.

Inicialmente, cada camada é analisada separadamente e as tarefas dentro de cada uma são particionadas de forma que sua atribuição seja a mais simples possível. Com isso, identificam-se aquelas que são comuns e que possam ser aproveitadas em diferentes trechos do código.

Interface entre Sistemas.

Determina-se então em quais componentes as bibliotecas de terceiros atuarão e qual a importância de cada biblioteca para o sistema final. Com isso, pode-se construir componentes que são interfaces (ou ainda, *wrappers*) para encapsular o uso de algumas dessas bibliotecas e assim facilitar a substituição da mesma se necessário.

A importância em se construir um encapsulamento em algumas das bibliotecas deve-se à facilidade da manutenção do código e à simplicidade em substituí-las ou substituir a implementação de determinados componentes, sem que isso impacte o sistema globalmente.

O aplicativo faz interface com vários sistemas desenvolvidos por terceiros, alguns deles proprietários. Os sistemas e os detalhes dessa integração estão descritos na Seção B.2.

Interação entre Componentes.

Com a definição de cada componente feita, resta estabelecer como cada um irá interagir com os outros. Para isso, separam-se os componentes de forma que fique clara a necessidade de cada um e, então, define-se uma maneira para que essa necessidade seja atendida por um outro componente. A Figura B.2 mostra os principais componentes do sistema e a interação entre cada um.

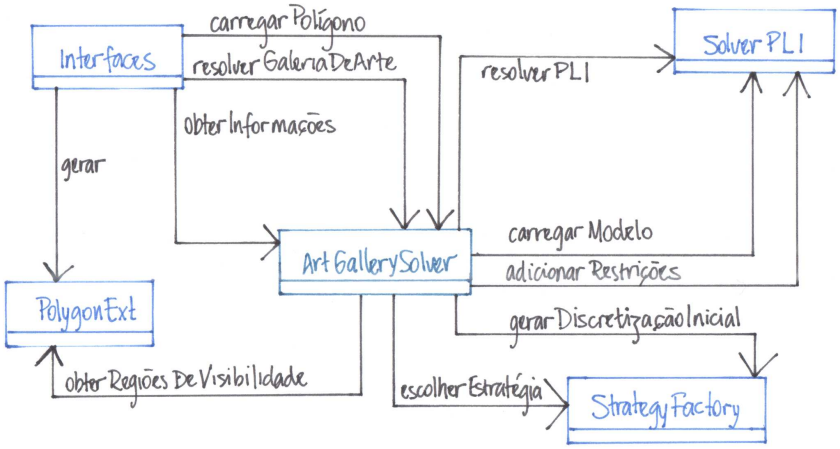


Figura B.2: Interação entre os principais componentes.

Note que as interações entre os componentes é feita de forma que a dependência de como o componente é implementado seja a menor possível. Isso fica claro ao observar-se, na Figura B.2, por exemplo, a definição de como são produzidas as estratégias de discretização. Repare que o sistema é independente de como os pontos da discretização são calculados e isso facilita a criação e o teste de novas estratégias sem impactar globalmente o aplicativo.

Separação em Pacotes.

Uma vez determinados os componentes, sua atribuição para a solução do problema e a interação entre eles, fica simples dividí-los em pacotes.

Isso é feito de forma a refletir muito proximamente a interface entre as camadas, entre as bibliotecas utilizadas e entre as atribuições em comum. A Figura B.3 mostra o diagrama de pacotes e a interação entre eles.

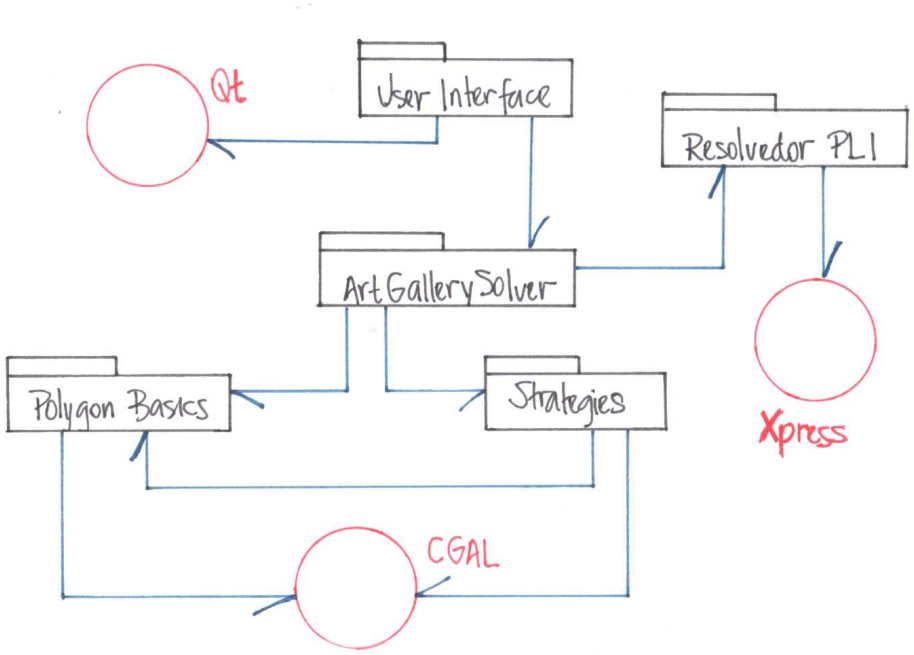


Figura B.3: Diagrama de pacotes simplificado.

Diagrama de Classe Simplificado.

Por fim, resta analisar cada um dos componentes e a atribuição designada a ele e dividí-lo em classes que a implementem de forma clara e concisa. A Figura B.4 exibe o diagrama simplificado de classes do sistema.

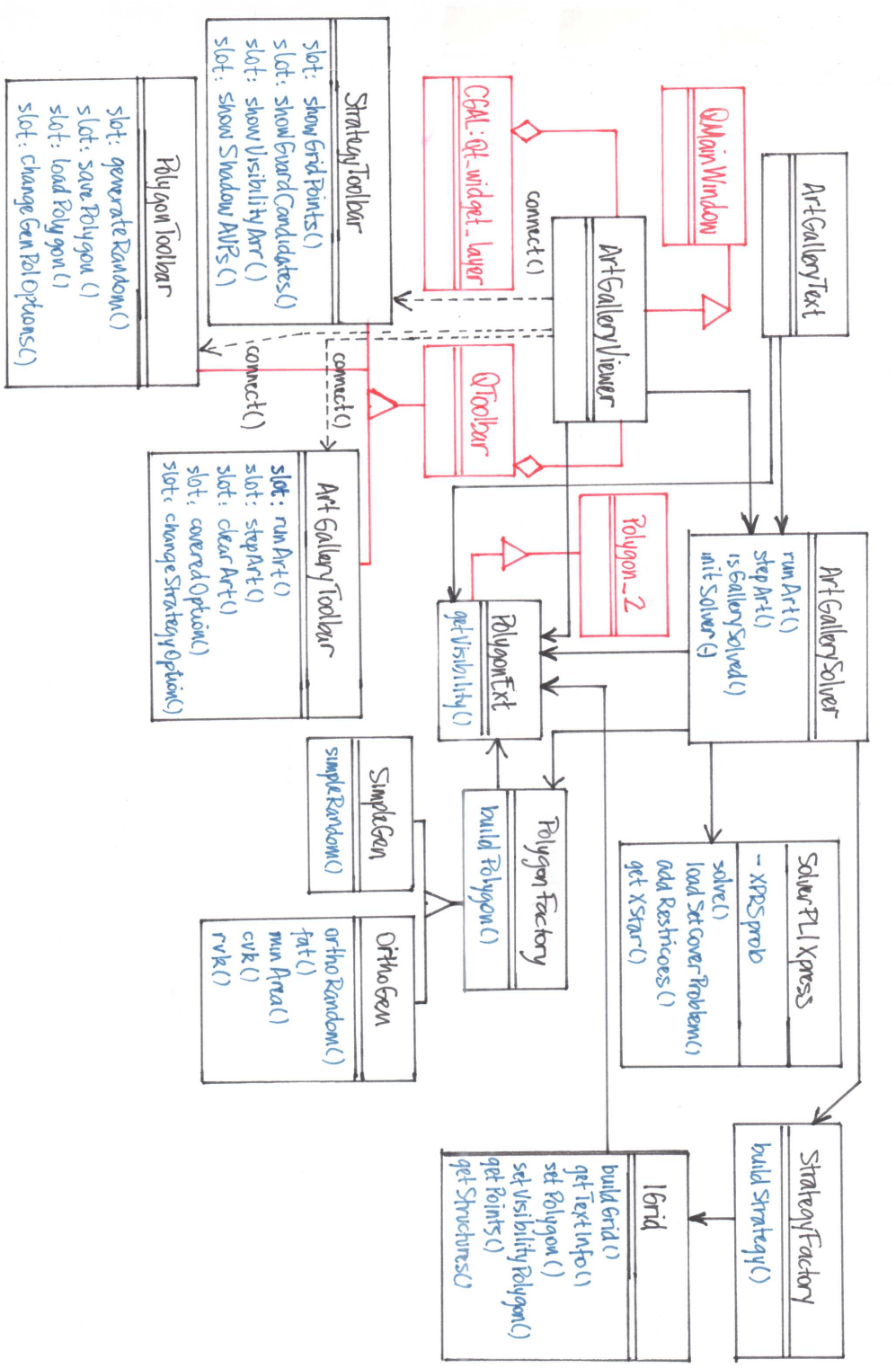


Figura B.4: Diagrama de classe simplificado.

B.2 Integração com *Software* de Terceiros

Com a definição do projeto e da arquitetura do sistema, ficam claros os pontos de integração com os *softwares* de terceiros.

O sistema foi desenvolvido utilizando várias bibliotecas que resolvem tarefas complexas ou provêem implementações de primitivas e algoritmos que auxiliam no desenvolvimento da aplicação como um todo. Foi também utilizado um *framework* para possibilitar que o sistema fosse independente de ambiente e um pacote para análise dos resultados obtidos.

A integração com cada um desses *softwares* está descrita a seguir.

B.2.1 *Framework* para Compilação

Para possibilitar que o sistema fosse independente de ambiente (compilador e sistema operacional), foi utilizado o *framework open-source* para compilação CMAKE [36].

Ele permite que o projeto seja compilado em qualquer ambiente, através de arquivos de configuração. O CMAKE apenas gerencia o processo de montagem do aplicativo, gerando arquivos intermediários (Makefiles no Unix e projetos/workspaces no Windows Visual C++ ou Eclipse CDT) para serem utilizados com o compilador do próprio ambiente, da forma usual.

Com isso, o sistema torna-se compatível com qualquer ambiente de desenvolvimento, não limitando sua utilização e evolução apenas a um grupo específico de desenvolvedores e pesquisadores.

B.2.2 Resolvedor Linear Inteiro

Como visto na Seção 3.1, o algoritmo faz a redução de uma instância do Problema de Galeria de Arte para uma instância do Problema de Cobertura de Conjuntos. Para solucionar este último, modela-se um Programa Linear Inteiro.

Dada a formulação PLI, utiliza-se um resolvedor linear inteiro para encontrar uma solução ótima para a instância. No trabalho apresentado nessa dissertação, utilizou-se o XPRESS [4] como resolvedor. Essa escolha deveu-se à grande eficiência apresentada pelo resolvedor, bem como à sua disponibilidade no laboratório.

Definição dos Valores dos Parâmetros.

A parametrização ideal para o problema foi escolhida após extensivos testes e está reproduzida abaixo.

- `XPRS_MAXTIME` = 100000.

  Limita o máximo tempo de uso de CPU para resolver uma instância pelo XPRESS, em segundos.

- `XPRS_PRESOLVEOPS` = 36863, ou seja, `1000 1111 1111 1111`.

  Habilita (bit setado para 1) e desabilita (bit em 0) as seguintes operações durante a etapa de *presolve*, onde cada bit representa:

  0 - Singleton column removal.
  1 - Singleton row removal.
  2 - Forcing row removal.
  3 - Dual reductions.
  4 - Redundant row removal.
  5 - Duplicate column removal.
  6 - Duplicate row removal.
  7 - Strong dual reductions.
  8 - Variable eliminations.
  9 - No IP reductions.
  10 - No semi-continuous variable detection.
  11 - No advanced IP reductions.
  14 - Linearly dependant row removal.
  15 - No integer variable and SOS detection.

- `XPRS_MIPPRESOLVE` = 0.

  Habilita (1) ou desabilita (0) o *presolve* do problema inteiro.

  Inicialmente, era utilizado um cálculo sobre a densidade de pontos da discretização para escolher se este parâmetro deveria estar habilitado ou não. Durante a evolução dos trabalhos, percebeu-se que era melhor deixá-lo desabilitado, pois não havia muita influência nas instâncias (de pequena densidade) na qual ele era mantido habilitado.

- `XPRS_PRESOLVE` = 0.

  Habilita (1) ou desabilita (0) o *presolve* de antes de começar a resolver o problema principal.

  Inicialmente esse parâmetro era setado da mesma forma como o anterior, hoje é mantido desabilitado.

- `XPRS_MIPLOG` = 3.

  Habilita a impressão de log do XPRESS para o modo mais verboso possível, facilitando assim a checagem intermediária dos resultados.

- `XPRS_MIPABSSTOP` = 0.9.

  Modifica a tolerância da melhor solução encontrada até o momento para parar com um gap menor ou igual a 0.9 da solução ótima.

- `XPRS_HEURSTRATEGY` = 2.

  Habilita o uso de heurísticas para um pouco além do básico e um pouco menos que uma abordagem extensiva.

- `XPRS_CUTSTRATEGY` = 2.

  Habilita uma estratégia de cortes moderada.

B.2.3 Biblioteca de Algoritmos Geométricos

Dada a natureza geométrica do problema, é necessária para sua resolução a implementação de vários algoritmos e primitivas geométricas.

Visto que o foco do trabalho não é realizar a implementação desses algoritmos, mas estudar a viabilidade prática do método proposto pela dissertação, utilizou-se a biblioteca geométrica e *open-source* CGAL [1], que contém a maior parte dos algoritmos necessários.

Essa biblioteca fornece suporte para diversos sistemas operacionais e oferece algoritmos e primitivas eficientes e confiáveis. Dentre os inúmeros pacotes existentes na biblioteca, os principais utilizados foram:

- `Kernel`, que contém estruturas básicas como pontos, segmentos, entre outras.
- `Gmpq`, que permite o uso de aritmética exata.
- `Polygons_2`, que provê o uso de algoritmos e operações booleanas realizadas em polígonos.
- `Arrangements_2`, que fornece suporte ao uso de arranjos, com base na estrutura de dados *Half-edge*.
- `GUI libraries Qt`, que encapsula o uso do *framework* para visualização gráfica, detalhado na próxima subseção.

B.2.4 *Framework* para Visualização Gráfica

Por se tratar de um problema geométrico, a visualização de dados e soluções é fundamental para uma melhor compreensão do Problema de Galeria de Arte. Nesse trabalho, optou-se pelo *framework* QT [2] para implementar essa visualização. Essa escolha levou em consideração que o QT é *open-source* e que está implementado em diversas plataformas como Windows, Linux e Mac. Ainda, está integrado à biblioteca de algoritmos geométricos utilizada, facilitando assim a codificação de uma visualização gráfica para os dados e os resultados. Assim, foi utilizado o conjunto de bibliotecas `GUI libraries Qt` do CGAL, que disponibiliza as ferramentas básicas necessárias para a visualização de componentes geométricos.

B.2.5 Pacote Estatístico

Para analisar com eficiência e robustez os resultados dos experimentos realizados, foi necessário recorrer a um pacote estatístico, que permitisse manipular uma larga quantidade de informação de maneira automatizada e que gerasse gráficos profissionais de maneira personalizada. Para isso, foi utilizado o R [3]. Trata-se de um pacote *open-source* que provê uma grande variedade de funções para manipulação de dados. Além de permitir uma fácil extensão das suas funcionalidades, ele apresenta um grande poder de personalização do resultado gráfico gerado, tudo mediante uma linguagem simples e própria de programação. Disponível para diversas plataformas e ambientes, o pacote estatístico mostrou-se essencial para organizar e apresentar de maneira rápida e eficiente os resultados que publicamos.

B.3 Extração e Processamento dos Resultados

Para avaliar experimentalmente o algoritmo proposto e comparar as diferentes estratégias de discretização apresentadas, várias métricas foram extraídas da execução do sistema. Esses dados podem ser agrupados em categorias diferentes, que apresentam informações relevantes sobre:

- **A instância**, incluindo dados como o identificador do arquivo, o tamanho do polígono e quantos candidatos a guarda existem.
- **A solução**, incluindo dados como o tamanho da solução gerada, a estratégia de discretização utilizada, quantidade de elementos que havia na discretização no momento da última iteração, quantidade de iterações necessárias para resolver a instância à otimalidade e tempo total gasto para isso.
- **Tempo de preprocessamento**, além de incluir dados como o tempo total, mostra esse tempo quebrado nas diversas fases do preprocessamento, como o tempo utilizado para o cálculo das regiões de visibilidades dos candidatos a guarda, o tempo de construção da discretização inicial, o tempo de geração do modelo PLI e o tempo de carga do modelo PLI.
- **Tempo de processamento**, além de exibir o tempo total gasto no processamento, ainda mostra como esse último é particionado entre o tempo utilizado para resolver o PLI, o tempo necessário para encontrar as áreas não cobertas pela solução atual, o tempo para encontrar os novos pontos para a discretização, o tempo para construir as novas restrições do modelo PLI e o tempo de inclusão dessas restrições no modelo.
- **A memória utilizada**, incluindo dados como a memória necessária para inicialização do modelo PLI, o quanto foi necessário para a computação da estratégia de discretização e o quanto foi necessário para resolver a instância à otimalidade.

Todos esse dados permitem analisar e comparar a eficiência do algoritmo e de cada uma das diferentes estratégias de discretização da galeria. Ainda, é possível estabelecer se o uso do algoritmo é viável na prática ou não.

No entanto, para que os resultados tenham significância estatística, é necessário que várias instâncias sejam geradas e analisadas. Como mostrado na Seção 2.4, analisar 30 instâncias para cada classe de polígono e quantidade de vértices mostrou-se suficiente para que o trabalho tivesse esse embasamento estatístico.

Nas análises efetuadas também foi feito um agrupamento dos dados de execução das instâncias, por classe de polígono, quantidade de vértices e estratégia de discretização utilizada, separando os *outliers*, ou seja, aquelas instâncias cujo tempo total de execução foi maior que uma vez e meio o desvio padrão da média. Vários *scripts* foram utilizados para montar esses agrupamentos e calcular estatísticas, a partir das quais, foram gerados os gráficos com as informações desejadas para a comparação das estratégias utilizadas.

B.4 Desafios de Implementação

Após a confecção da arquitetura e a definição das atribuições de cada classe, teve início a implementação do aplicativo. Durante essa fase, surgiram diversos desafios que foram transpostos ou contornados. Alguns desses desafios, os mais importantes, são discutidos a seguir.

B.4.1 Região de Visibilidade

Uma grande quantidade de desafios surgiram durante a busca por um algoritmo que construísse uma região de visibilidade corretamente, principalmente devido ao fato de que, ao trabalhar-se com guardas estacionários em vértices e com as instâncias descritas na Seção 2.4, vários casos degenerados ocorrem.

Inicialmente, foram pesquisadas algumas implementações disponíveis em bibliotecas geométricas, mas nenhuma delas tratava corretamente os casos degenerados. Por fim, recorreu-se à literatura, onde foram sucessivamente propostos alguns algoritmos com complexidade linear [30, 31, 33], até que um deles tornou-se confiável. No entanto, ao elevar o número de vértices, o algoritmo não se mostrou robusto na prática, falhando com freqüência na construção de determinadas regiões de visibilidade. Isso ocorreu devido ao emprego pelo algoritmo de operações não algébricas, cujos resultados dependem da precisão aritmética utilizada. Assim, o algoritmo fica sujeito a problemas de aproximação, arredondamento e precisão, que se acumulam quando se utilizam ângulos em longas sequências de cálculos.

Para resolver esse problema, foram pesquisados além de outros algoritmos para o

cálculo da região de visibilidade, formas de evitar o uso de operações não algébricas. Isso conduziu à modificação do algoritmo [30] com a introdução da técnica de pseudo-ângulos, a qual extinguiu a necessidade do uso de tais operações, tornando o cálculo das regiões de visibilidade correto, preciso e robusto.

B.4.2 Localização de Ponto dentro de Polígono Estrelado

Durante a implementação do algoritmo, percebeu-se que um tempo não desprezível era despendido para determinar se um ponto da discretização fazia parte da região de visibilidade de determinado candidato a guarda. O algoritmo utilizado para executar essa tarefa era aquele disponibilizado no CGAL que tem complexidade $O(n)$, visto que ele não explora qualquer conhecimento mais amplo do tipo de polígono com o qual está trabalhando.

Sabendo que a região de visibilidade de um ponto é um polígono estrelado, pode-se utilizar um algoritmo [40] que consome tempo $O(\log n)$ para esse mesmo cálculo, fazendo simplesmente uma busca binária. No entanto, experimentalmente, o algoritmo desenvolvido para essa busca não superou a eficiência daquele implementado no CGAL que, embora leve tempo linear no pior caso, possui um número médio de operações – para o tamanho das instâncias trabalhadas – menor que o outro algoritmo, que tem que construir estruturas auxiliares mais complexas.

B.4.3 Arranjo de Visibilidade

Durante a computação de determinadas estratégias, faz-se necessário a geração do arranjo de visibilidade, composto pelas arestas do polígono e por todas as arestas das regiões de visibilidade dos candidatos a guardas.

Para a construção desse arranjo, foram pesquisadas algumas técnicas e algoritmos que poderiam se encaixar melhor nos objetivos desse trabalho. Inicialmente, foi decidido construir o arranjo com uma estrutura criada sobre *half-edges* [35]. A partir do polígono original, adiciona-se uma a uma as arestas da região de visibilidade, atualizando e mantendo essa estrutura consistente. No entanto, devido a sobre utilização de objetos, classes, e checagens, não se chegou a uma implementação do algoritmo com a eficiência esperada.

Após alguma pesquisa, optou-se por trabalhar com o modelo de `Arrangement_2` do CGAL, construído sobre a estrutura de *half-edge* e com um observador, que permitia a computação de valores para serem adicionados à estrutura, de forma que determinadas propriedades pudessem ser construídas e recuperadas mais tarde. Com essa implementação, a eficiência da construção do arranjo de visibilidade aumentou, e passou a ser factível a resolução de instâncias com um maior número de vértices.

B.4.4 Instâncias Degeneradas

Por ser um problema prático, o algoritmo deve ser robusto o suficiente para lidar com todos os tipos de instâncias. O desafio é que a maior parte das instâncias, em especial as aleatórias ortogonais e as de von Koch, são degeneradas, isto é, possuem inúmeros vértices colineares. Desse modo, uma importante parte do trabalho foi adaptar determinados algoritmos para lidar corretamente com essas instâncias, em especial a adaptação do algoritmo que realiza a construção da região de visibilidade de um ponto. Foi feito um estudo minucioso desse algoritmo e levantados diversos casos onde o mesmo falhava (compare o exemplo da Figura B.5 com o da Figura B.6).

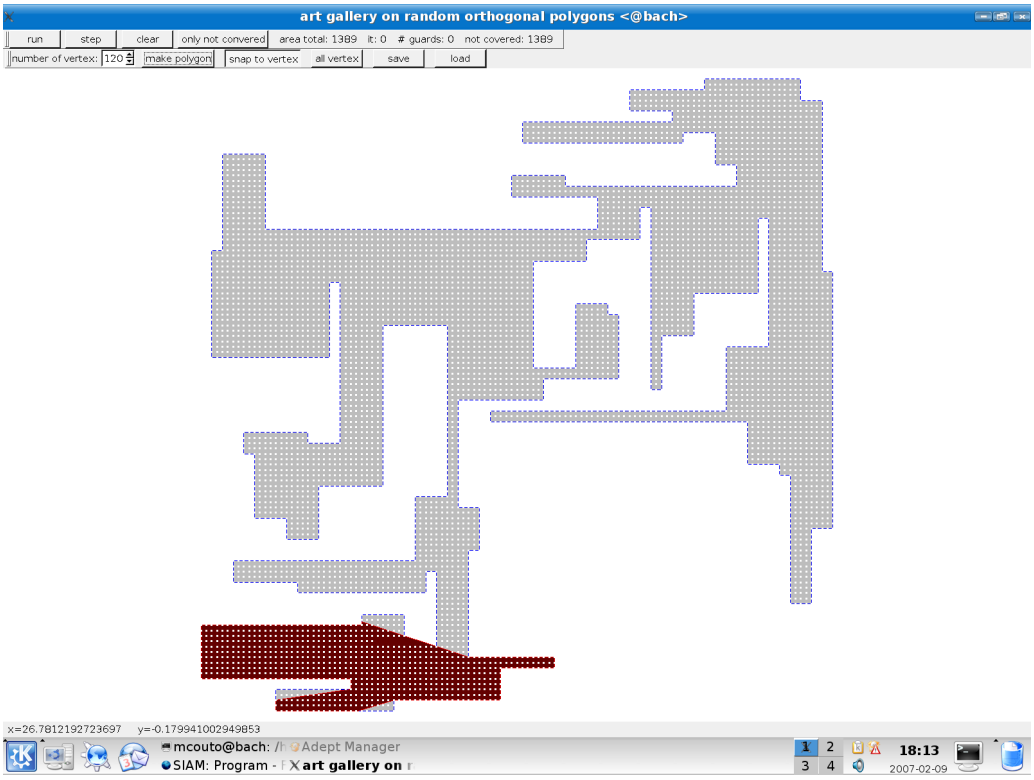


Figura B.5: Falha no algoritmo de visibilidade devido a casos degenerados.

Através de um cuidadoso processo de análise, fez-se algumas mudanças pontuais no código que permitiram que esses casos fossem solucionados corretamente, gerando resultados consistentes, como pode ser visto no mesmo exemplo anterior, mas agora corrigido, na Figura B.6.

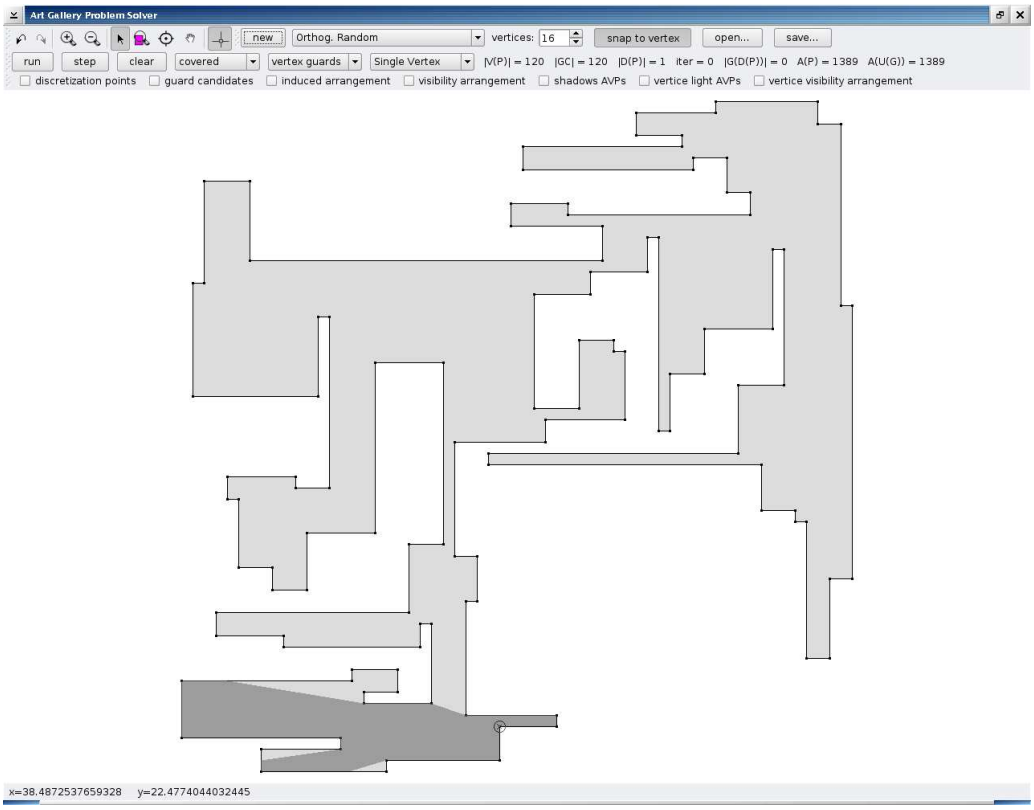


Figura B.6: Falha corrigida no algoritmo de visibilidade.

B.4.5 Aritmética Exata

Um dos primeiros e mais essenciais pontos a serem considerados ao se desenvolver aplicativos geométricos é a forma como será tratada a aritmética do projeto. Aqui, vários aspectos da implementação e dos objetivos a serem alcançados devem ser ponderados, de modo que o resultado seja tão robusto e eficiente como esperado. Note que um pequeno erro de aproximação pode fazer com que todo o resultado seja interpretado erroneamente.

Para o algoritmo implementado nessa dissertação, foi escolhida uma aritmética que fosse exata e robusta nos termos apresentados e, ao mesmo tempo, fosse eficiente. Utilizou-se para esse fim a biblioteca GMP, do inglês *GNU Multiple-Precision Library*. Essa biblioteca pode ser utilizada como parâmetro para os *templates* da biblioteca de algoritmos geométricos. Além de ser gratuita, ela trabalha com aritmética de precisão arbitrária, onde o limite prático da sua precisão é a quantidade de memória disponível na máquina.

A GMP é tida como uma das mais rápidas bibliotecas que trabalham com grandes valores de números pois, entre outros fatores, utiliza diferentes algoritmos para diferentes tamanhos de operandos, além de possuir código otimizado para diferentes processadores.

É interessante notar que os principais erros encontrados durante o desenvolvimento

do aplicativo foram os erros referentes ao uso de aritmética inexata. O maior deles foi a utilização de operações não algébricas para o cálculo da região de visibilidade de um ponto. Esse problema acabou contornado com o uso da técnica de pseudo-ângulos.

B.5 Interfaces

Devido à natureza geométrica do Problema de Galeria de Arte e ao caráter prático da experimentação desejada, é natural que fosse desenvolvida ao menos uma interface para a aplicação. Aliada à necessidade de visualização dos resultados e de detalhes da solução está também a necessidade de executar um grande número de instâncias de forma automática, permitindo assim que dados importantes sejam extraídos rapidamente.

Decidiu-se, portanto, pela criação de duas interfaces complementares, uma para a visualização completa do problema, possibilitando diversas análises detalhadas e outra possibilitando a geração de *scripts* para a execução de baterias de testes e coleta de grande quantidade de informações relevantes.

B.5.1 Texto (Linha de Comando)

Para viabilizar a execução automatizada de uma grande quantidade de instâncias, de maneira a onerar o menos possível o ambiente e garantir que o tempo de execução seja apenas da resolução do problema, foi criada uma interface de linha de comando. A Figura B.7 exibe a interface desenvolvida e todos os parâmetros que podem ser passados para a mesma.

```
./artGalleryText [-gen <type> <size> | -load <file>] [-id <idFile>]
                 [-outPolFile <file>] [-logFile <file>]
                 [-grid <gridType>] [-solve]
```

onde,

```
type:      <RANDOM | FAT | MIN | SIMPLE | VON | RANDVON>
gridType:  <SINGLE | REGULAR | ALL_VERTICES | INDUCED | SHADOW_AVPS |
            COMPLETE_AVPS | CONVEX>
```

Figura B.7: Interface por Linha de Comando do Aplicativo.

Note, que é possível gerar instâncias de diversos tipos e tamanhos (`-gen`) ou carregá-la de um arquivo (`-load`). Ainda, é possível escolher o tipo de estratégia da discretização inicial que será utilizada (`-grid`) e, por fim, pode-se definir se o polígono utilizado na execução será salvo em um arquivo (`-outPolFile`) e se o resultado será guardado em um outro (`-logFile`).

Os outros parâmetros da interface servem para identificar a instância nos resultados exibidos (`-id`) e informar se é desejado que a instância seja resolvida à otimalidade (`-solve`).

B.5.2 Visualização (Gráfica)

Um dos diferenciais de aplicativos geométricos é a possibilidade da visualização da solução do problema e de como ela foi encontrada. Ao desenvolver uma interface gráfica, permite-se que determinados aspectos do problema sejam melhor entendidos, possibilitando acelerar o aprendizado sobre o mesmo.

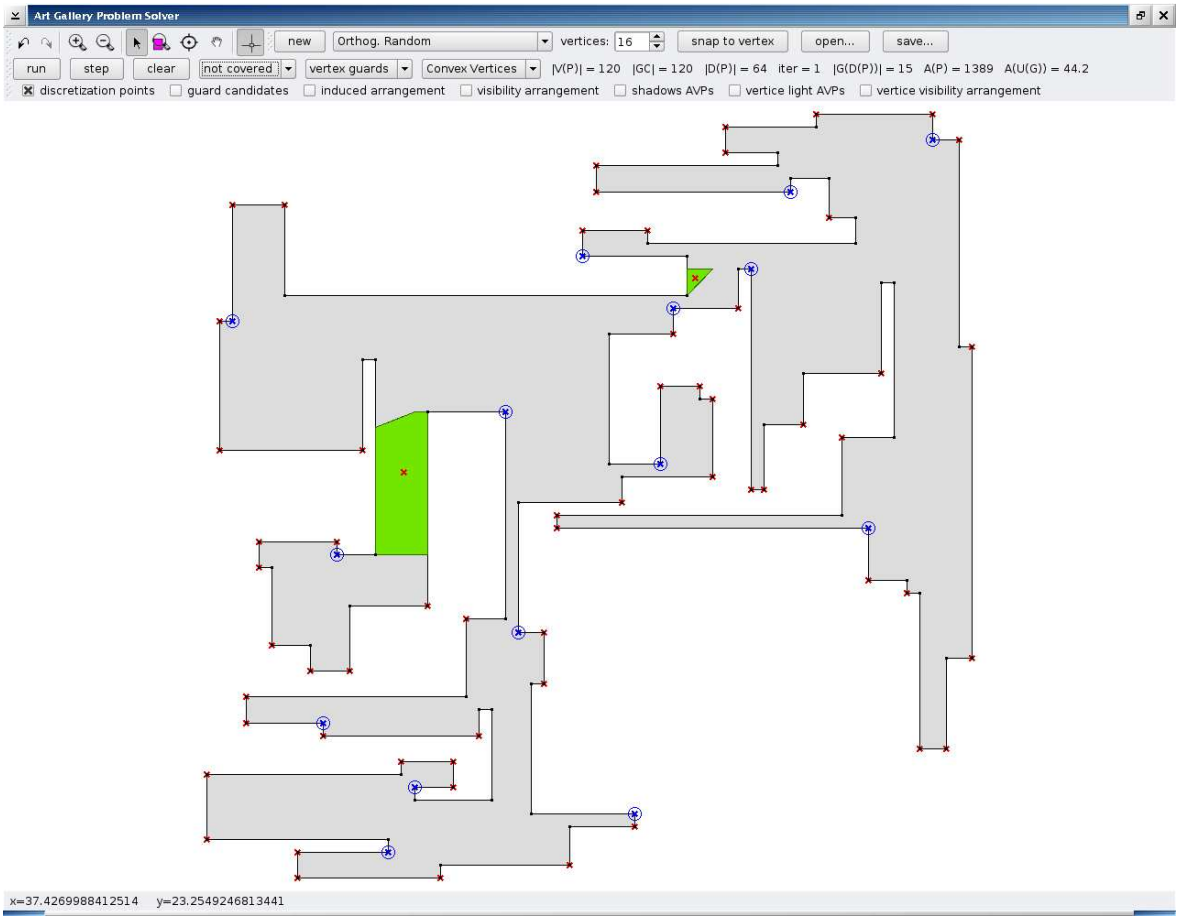


Figura B.8: Interface Visual do Aplicativo.

A interface gráfica mostrou-se extremamente útil para enfrentar alguns dos desafios discutidos na Seção B.4. Com ela, foi possível identificar e propor, com maior velocidade, soluções para determinadas dificuldades. Além disso, tornou os testes do algoritmo relativamente mais simples, rápidos e fáceis de serem realizados.

A visualização do processo de encontrar uma solução fez com que novas idéias surgissem, melhorando assim o trabalho reportado nessa dissertação. A Figura B.8 mostra um exemplo de uma tela da versão atual da interface desenvolvida. Repare que, aparte da região onde são exibidas as informações correntes da busca de uma solução para a instância, para uma melhor interação com o usuário, a interface provê diferentes funcionalidades através de uma barras de tarefas. Cada uma delas é responsável por uma parte específica do processo de obtenção da solução, conforme descrito em maior profundidade a seguir.

Barra de Tarefas CGAL.

A barra de tarefas CGAL foi a primeira a ser inserida na interface. Posto que era um componente pronto, inerente às classes utilizadas, ela foi acrescentada ao aplicativo com um mínimo de esforço. Ela provê importantes ferramentas, que auxiliam na visualização de detalhes, como o *zoom in/out*, histórico de movimentos, coordenada do cursor, centralização de objetos, entre outras.

Barra de Tarefas Polígono.

A barra de tarefas Polígono permite um controle do usuário sobre as instâncias que estão sendo utilizadas. Através dela, ele pode gerar uma nova instância – de um tipo e com uma quantidade de vértices específicos – ou ainda salvar a instância atual ou carregar uma instância salva, produzida anteriormente.

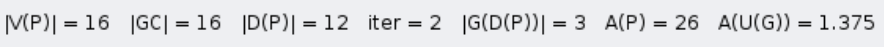
O usuário consegue também controlar se ao pressionar o botão direito do *mouse* a região de visibilidade exibida será aquela do ponto onde ele clicou ou se será aquela do vértice mais próximo ao ponto clicado.

Barra de Tarefas Galeria de Arte.

Esta é a barra mais importante do aplicativo, pois contém toda a interface para a interação do usuário com a solução do problema. Nela, ele pode escolher se deseja resolver o problema e visualizar passo-a-passo esse processo, ou se deseja apenas visualizar a solução final.

Além dessa escolha, o usuário pode determinar qual a estratégia de discretização inicial que deve ser aplicada à instância e se ele deseja ver, durante as iterações, a área momentaneamente coberta, descoberta ou apenas a instância e os guardas atuais.

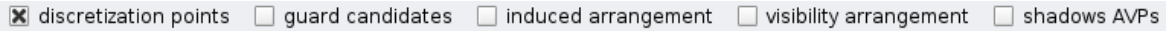
Barra de Tarefas Informações.



A barra de informações mostra os dados momentâneos da solução da instância. Nela, pode-se observar:

- `|V(P)|`
  Quantidade de vértices da instância.
- `|GC|`
  Quantidade de pontos candidatos a guarda.
- `|D(P)|`
  Quantidade de pontos existentes na discretização atual da instância.
- `iter`
  Número da iteração atual.
- `|G(D(P))|`
  Quantidade mínima de guardas exigidos para cobrir os pontos da discretização atual.
- `A(P)`
  Área interna da instância.
- `A(U(G))`
  Área interna da instância não coberta pelo conjunto de guardas atual.

Barra de Tarefas Estratégias.



Por fim, tem-se a barra de tarefas que gerencia a exibição de informações inerentes às estruturas geradas por algumas das estratégias. Assim, é possível visualizar os pontos atuais da discretização da instância; os pontos que são candidatos a guarda; qual o arranjo induzido da instância; qual o arranjo de visibilidade dos pontos candidatos a guarda e, quais são os AVPs de sombra existentes naquela instância.

Referências Bibliográficas

[1] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[2] QT. http://qt.nokia.com.

[3] R, The R Project for Statistical Computing. http://www.r-project.org.

[4] XPRESS Optimizer. http://www.dashopt.com.

[5] A. Aggarwal, S. K. Ghosh, and R. K. Shyamasundar. Computational complexity of restricted polygon decompositions. In G. T. Toussaint, editor, *Computational Morphology*, pages 1–11. North-Holland, Amsterdam, Netherlands, 1988.

[6] Y. Amit, J. S. B. Mitchell, and E. Packer. Locating guards for visibility coverage of polygons. In *Proc. Workshop on Algorithm Eng. and Experiments*, pages 1–15, 2007.

[7] D. Avis and G. T. Toussaint. An optimal algorithm for determining the visibility of a polygon from an edge. *IEEE Trans. Comput.*, C-30(12):910–1014, 1981.

[8] T. Baumgartner, S. Fekete, A. Kröller, and C. Schmidt. Exact solutions and bounds for general art gallery problems. In *Proc. of 12th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 11–22, 2010.

[9] P. Bose, A. Lubiw, and J. I. Munro. Efficient visibility queries in simple polygons. *Computational Geometry*, 23(3):313–335, 2002.

[10] V. Chvátal. A combinatorial theorem in plane geometry. In *Journal of Combinatorial Theory Series B*, volume 18, pages 39–41, 1975.

[11] M. C. Couto, P. J. de Rezende, and C. C. de Souza. An exact algorithm for an art gallery problem. Technical Report IC-09-46, Institute of Computing, University of Campinas, November 2009. In English, 25 pages.

[12] M. C. Couto, P. J. de Rezende, and C. C. de Souza. An ip solution to the art gallery problem. In *SCG '09: Proceedings of the 25th annual symposium on Computational geometry*, pages 88–89, New York, NY, USA, 2009. ACM.

[13] M. C. Couto, P. J. de Rezende, and C. C. de Souza. Video: An ip solution to the art gallery problem. In *18th Annual Video/Multimedia Review of Computational Geometry as part of the 25th annual symposium on Computational geometry*, pages 88–89, New York, NY, USA, 2009. ACM.

[14] M. C. Couto, C. C. de Souza, and P. J. de Rezende. An exact and efficient algorithm for the orthogonal art gallery problem. In *Proc. of the XX Brazilian Symp. on Comp. Graphics and Image Processing*, pages 87–94. IEEE Computer Society, 2007.

[15] M. C. Couto, C. C. de Souza, and P. J. de Rezende. Experimental evaluation of an exact algorithm for the orthogonal art gallery problem. In *WEA*, volume 5038 of *Lecture Notes in Computer Science*, pages 101–113. Springer, 2008.

[16] M. C. Couto, C. C. de Souza, and P. J. de Rezende. Strategies for optimal placement of surveillance cameras in art galleries. In *GraphiCon 2008: XI International Conference on Computer Graphics & Vision*, volume 1, page http://www.graphicon.ru/2008/proceedings/technical.html. Lomonosov Moscow State University, 2008.

[17] M. C. Couto, C. C. de Souza, and P. J. de Rezende. Instances for the Art Gallery Problem, 2009. *www.ic.unicamp.br/∼cid/Problem-instances/Art-Gallery*.

[18] J. Culberson and R. Reckhow. Dent diagrams: A unified approach to polygon covering problems. Technical Report TR 87-14, Dept. Comput. Sci., Univ. Alberta, Edmonton, Alberta, Canada, 1987.

[19] J. Culberson and R. A. Reckhow. Covering a simple orthogonal polygon with a minimum number of orthogonally convex polygons. In *Proc. 3rd Annu. ACM Sympos. Comput. Geom.*, pages 268–277, 1987.

[20] L. H. de Figueiredo and P. C. P. Carvalho. *Introdução à Geometria Computacional*. Instituto de Matemática Pura e Aplicada, 1991.

[21] S. Eidenbenz. Approximation algorithms for terrain guarding. *Inf. Process. Lett.*, 82(2):99–105, 2002.

[22] U. M. Erdem and S. Sclaroff. Optimal placement of cameras in floorplans to satisfy task requirements and cost constraints. In *Proc. International Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras*, pages 30–41, 2004.

[23] U. M. Erdem and S. Sclaroff. Automated camera layout to satisfy task-specific and floor plan-specific coverage requirements. *Comput. Vis. Image Underst.*, 103(3):156–169, 2006.

[24] V. Estivill-Castro, J. O'Rourke, J. Urrutia, and D. Xu. Illumination of polygons with vertex floodlights. *Inform. Process. Lett.*, 56:9–13, 1995.

[25] K. Falconer. *Fractal Geometry, Mathematical Foundations and Applications*. John Wiley & Sons, 1990. pp. 120–121.

[26] S. Fisk. A short proof of Chvátal's watchman theorem. In *Journal of Combinatorial Theory Series B*, volume 24, page 374, 1978.

[27] S. K. Ghosh. Approximation algorithms for art gallery problems in polygons. *Discrete Applied Mathematics*, 158(6):718–722, 2010.

[28] F. Hausdorff. Dimension und äußeres maß. *Mathematische Annalen*, 79(1):157–179, 1918.

[29] R. Honsberger. *Mathematical Gems II*. Number 2 in The Dolciani Mathematical Expositions. Mathematical Association of America, 1976.

[30] B. Joe and R. B. Simpson. Visibility of a simple polygon from a point. Report CS-85-38, Dept. Math. Comput. Sci., Drexel Univ., Philadelphia, PA, 1985.

[31] B. Joe and R. B. Simpson. Correction to Lee's visibility polygon algorithm. *BIT*, 27:458–473, 1987.

[32] J. Kahn, M. M. Klawe, and D. Kleitman. Traditional galleries require fewer watchmen. *SIAM J. Algebraic Discrete Methods*, 4:194–206, 1983.

[33] D. T. Lee. Visibility of a simple polygon. *Comput. Vision, Graphics, and Image Process*, 22:207–221, 1983.

[34] D. T. Lee and A. K. Lin. Computational complexity of art gallery problems. *IEEE Trans. Inf. Theor.*, 32(2):276–282, 1986.

[35] M. Mäntylä. *An Introduction to Solid Modeling*. Computer Science Press, Rockville, MD, 1988.

[36] K. Martin and B. Hoffman. *Mastering CMake: A Cross-Platform Build System*. Kitware, Inc., 2003.

[37] J. O'Rourke. Galleries need fewer mobile guards: a variation on Chvátal's theorem. *Geom. Dedicata*, 14:273–283, 1983.

[38] J. O'Rourke. *Art Gallery Theorems and Algorithms*. Oxford University Press, 1987.

[39] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[40] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 3rd edition, Oct. 1990.

[41] D. Schuchardt and H.-D. Hecker. Two NP-hard art-gallery problems for ortho-polygons. *Mathematical Logic Quarterly*, 41:261–267, 1995.

[42] T. C. Shermer. Recent results in art galleries. *Proceedings of the IEEE*, 80(9):1384–1399, 1992.

[43] A. P. Tomás and A. L. Bajuelos. Generating random orthogonal polygons. In *Current Topics in Artificial Intelligence*, volume 3040 of *Lecture Notes in Computer Science*, pages 364–373. Springer Berlin / Heidelberg, 2004.

[44] A. P. Tomás, A. L. Bajuelos, and F. Marques. On visibility problems in the plane - solving minimum vertex guard problems by successive approximations. In *Proceedings of the 9th International Symposium on Artificial Intelligence and Mathematics (AI & MATH 2006)*, 2006. to appear.

[45] J. Urrutia. Art gallery and illumination problems. In J.-R. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 973–1027. North-Holland, 2000.