UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

MAC0215: Relatório Final
O Problema da Visita de Polígonos

Gabriel Freire Ushijima

São Paulo, SP
7 de novembro de 2025

1 Introdução

Esse projeto de pesquisa teve como objetivo explorar o Problema da Visita de Polígonos (Touring Polygons Problem - TPP), um problema de otimização geométrica que envolve encontrar o caminho mais curto que visita uma sequência de polígonos no plano. O projeto foi desenvolvido ao longo do semestre como parte da disciplina MAC0215 - Atividade Curricular em Pesquisa (2025), sendo supervisionado pelo Prof. Dr. Ernesto G. Birgin.

Nesse relatório, vamos apresentar o problema que estamos resolvendo, discutir a base bibliográfica que fundamenta nosso trabalho, detalhar as implementações realizadas, analisar os resultados obtidos e propor possíveis melhorias e trabalhos futuros. Também vamos dedicar uma seção para discutir como o tempo foi alocado ao longo do projeto, justificando o investimento de ao menos 100 horas de trabalho.

Todo o código desenvolvido para esse projeto está disponível no repositório do GitHub, onde também é possível encontrar instruções para executar os algoritmos implementados, assim como acompanhar o desenvolvimento gradual do projeto, servindo como evidência do trabalho realizado.

2 Base Bibliográfica

Esse problema não é original e foi apresentado pela primeira vez por Dror et al. (2003) [1]. No entanto, desde então não houve muitos avanços significativos na resolução do problema, e a maioria das soluções propostas ainda se baseia nas ideias apresentadas nesse artigo inicial.

Enquanto a tarefa de implementar os algoritmos propostos no artigo de Dror et al. (2003) [1] pode parecer direta, na prática, a implementação desses algoritmos apresenta desafios significativos, uma vez que o artigo em si é bastante denso e complexo, focando mais na análise teórica e demonstração matemática de corretudo do que na implementação prática, tanto que não incluem pseudocódigo, visualizações ou detalhes de implementação.

Por esse motivo, esse projeto propôs desbravar esse problema e artigo, buscando entender as ideias apresentadas e desenvolver implementações práticas dos algoritmos propostos. Dessa forma, podemos reapresentar soluções concretas para o problema, contribuindo para a literatura existente e abrindo caminho para futuras pesquisas e melhorias na área.

Além disso, também propomos variações do problema original, não cobertas na literatura atual, explorando diferentes restrições e características dos polígonos envolvidos, o que nos permite ampliar o escopo do estudo e contribuir com novas perspectivas para o campo.

3 O Problema

O Problema da Visita de Polígonos (TPP) consiste em encontrar o caminho mais curto que visita um conjunto de polígonos no plano. Esse problema é um caso específico do problema do caixeiro viajante com vizinhanças (TSPN) [2], onde o objetivo é visitar regiões genéricas no plano, que é por sua vez uma generalização do problema do caixeiro viajante (TSP) [3], onde o objetivo é visitar pontos específicos.

O enunciado formal do TPP é o seguinte: dado um ponto inicial s, um ponto final t e uma sequência de polígonos $P_1, P_2, \ldots, P_k$ no plano, encontrar o menor caminho que começa em s, termina em t e visita cada polígono P_i pelo menos uma vez, além disso os polígono devem ser tocados na ordem que são dados. Consideramos que visitar um polígono significa que o caminho pode atravessar ou simplesmente tocar na borda de cada polígono.

Para esse projeto, consideramos três variações do problema, o TPP Irrestrito, onde lidamos com polígonos convexos e disjuntos, o TPP Restrito onde os polígonos ainda devem ser convexos, mas pode ter interseção e consideramos a inclusão de uma "cerca"que restringe nosso movimento e o TPP com polígonos não convexos, onde retornamos ao TPP Irrestrito, mas consideramos que os polígonos pode ser não convexos. Vamos discutir cada um deles nas próximas seções.

3.1 TPP Irrestrito

Essa é a primeira variação do problema que abordamos, vamos considerar que recebemos um ponto inicial s, um ponto final t e uma sequência de polígonos $P_1, P_2, \ldots, P_k$ e buscamos um caminho mínimo que enconsta em cada polígono no plano, como no enunciado geral do problema. Além dessas suposições, vamos considerar que os polígonos são **convexos** e são **disjuntos**.

A figura ao lado 1 ilustra um exemplo de entrada e solução para o problema. O caminho mínimo é representado pela linha roxa, que começa no ponto s (ponto verde), termina no ponto t (ponto vermelho) e toca o triângulo, trapézio e então pentágono.

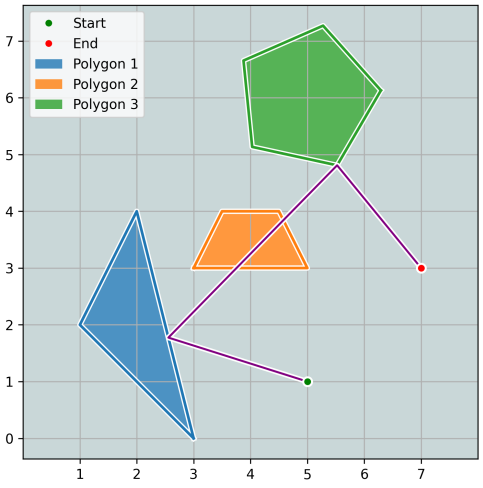

Figura 1: Caminho mínimo para um caso de 3 polígonos.

O algoritmo que implementamos é descrito no artigo [1] como tendo complexidade $O(nk\log(n/k))$, onde n é o número total de vértices dos polígonos e k é o número de polígonos. Os autores não descreveram as estruturas de dados nem detalhes que facilitassem a implementação, então tivemos que fazer diversas escolhas de implementação para conseguir a complexidade prometida.

A solução em si se baseia na ideia de, para cada polígono P_i, particionar o plano em três tipos de região, permitindo que façamos consultas do tipo $q = \text{Query}(p, i)$, onde q é o último passo do menor caminho que parte de s, toca cada polígono $P_1, \ldots, P_i$ e chega em p. Formulando o problema dessa maneira, podemos observar que, se conseguirmos criar todas as partições para cada polígono, podemos resolver o problema facilmente, definindo $q_k = \text{Query}(t, k)$, então $q_{k-1} = \text{Query}(q_k, k-1)$, e assim por diante, até chegarmos em $q_0 = s$. Dessa forma, conseguimos reconstruir o caminho mínimo de trás para frente.

Na figura abaixo 2 é possível ver um exemplo dessas partições, para polígono criamos uma partição diferente, usando as partições anteriores. Dessa forma a primeira partição (a do triângulo) usa apenas o ponto s, a segunda partição (a do trapézio) usa a partição do triângulo e s, e a terceira partição (a do pentágono) usa a partição do triângulo, trapézio e s.

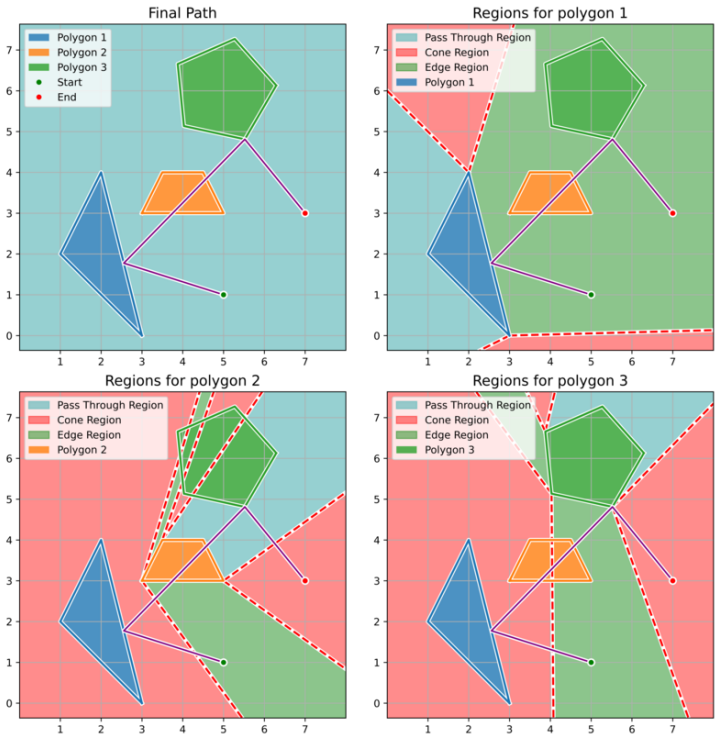

Figura 2: Partições para um caso de 3 polígonos.

Essas regiões são de três tipos, chamdas de regiões de vértice (em vermelho), aresta (em verde) e sombra (em ciano). As regiões de vértice são associadas à um vértice específico do polígono P_i, e consistem em todos os pontos p tais que o caminho mínimo toca em P_i nesse vértice e então chega em p. As regiões de aresta são associadas a uma aresta específica do polígono P_i, e consistem em todos os pontos p tais que o caminho mínimo toca em P_i nessa aresta e então chega em p. Finalmente, as regiões de sombra são associadas ao polígono em si e consistem em todos os pontos p tais que o caminho mínimo toca em atravessa P_i para chegar em p.

A seguir vamos construir o caminho mínimo para o exemplo da figura 1, usando as partições da figura 2. Começamos com $q_3 = \text{Query}(t, 3)$, olhando na partição do pentágono, observamos que t está na região de vértice, assim, sabemos que q_3 deve ser esse vértice de P_3. Em seguida, calculamos $q_2 = \text{Query}(q_3, 2)$, olhando na partição do trapézio, observamos que q_3 está na região de sombra, assim, podemos essencialmente ignorar P_2, pois qualquer ponto mínimo até q_3 que toque P_2 deve necessariamente atravessá-lo. Finalmente, calculamos $q_1 = \text{Query}(q_3, 1)$, olhando na partição do triângulo, observamos que q_3 está na região de aresta, assim, sabemos que q_1 deve ser o ponto na aresta de P_1 que minimiza a distância de s até q_2. Com isso, conseguimos reconstruir o caminho mínimo completo.

Tendo essa ideia geral em mente, resta apenas como desafio criar essas partições e localizar pontos nelas de forma eficiente. Essencialmente, para criar a partição de P_i, devemos calcular um caminho mínimo

que toca cada polígono $P_1, \ldots, P_{i-1}$ até cada vértice de P_i e apartir da maneira que esse caminho chega nesses vértices, podemos determinar as regiões de vértice. Uma vez que temos as regiões de vértice, notamos que as regiões de aresta e sombra são as regiões restantes no plano, dessa forma, é apenas necessário saber diferenciar essas regiões. Para isso, podemos observar que se um caminho mínimo que chega em um ponto interno de uma aresta de P_i atravessa P_i, essa aresta deve pertencer à região de sombra, caso contrário, pertence à região de aresta. Com isso, conseguimos construir todas as partições necessárias.

É claro que essa explicação é bastante simplificada, e muitos detalhes de implementação foram omitidos para manter o foco na ideia geral. O artigo original [1] apresenta todas os lemas e teoremas necessários para garantir a corretude do algoritmo e a complexidade prometida, enquanto temos uma relatório explicando os detalhes de implementação e o código em si por completo no repositório do GitHub associado a esse projeto.

3.2 TPP Restrito

Essa variação do problema faz as mesmas suposições do TPP Irrestrito, mas agora permitimos que os polígonos possam se intersectar. Além disso, introduzimos como entrada "cercas" $F_0, \ldots, F_k$ tais que F_i contém P_i e P_{i+1} para todo $0 \leq i \leq k$, considerando $P_0 = s$ e $P_{k+1} = t$, e entre o caminhoo visitar P_i e P_{i+1}, ele deve permanecer dentro de F_i.

Essas cercas devem ser polígonos, mas não necessariamente convexos, na verdade, uma cerca convexa é equivalente a não ter cerca alguma, então faz sentido permitir que as cercas sejam não convexas. A figura ao lado 3 ilustra um exemplo de entrada e solução para esse problema, com dois polígons e três cercas. Note que o menor caminho (linha roxa) deve permanecer dentro das cercas (linhas coloridas) ao visitar os polígonos (polígonos preenchidos), criando um caminho diferente do TPP Irrestrito.

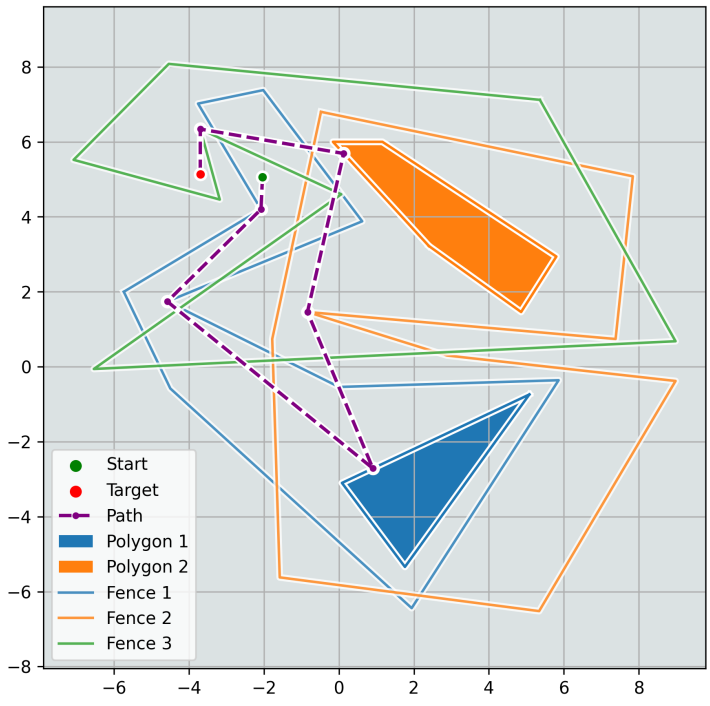

Figura 3: Caminho mínimo para um caso de 2 polígonos e 3 cercas.

O tratamento desse problema no artigo [1] é ainda mais sucinto do que o TPP Irrestrito, com pouco detalhes e nenhuma visualização do problema, sendo uma leitura extremamente densa e teórica, sem nenhum detalhe de implementação. No entanto, a ideia geral do algoritmo é semelhante ao TPP Irrestrito, onde criamos partições do plano para cada polígono P_i e usamos essas partições para reconstruir o caminho mínimo. A principal diferença é que, ao criar as partições, devemos considerar as cercas F_i, garantindo que o caminho mínimo permaneça dentro dessas cercas ao visitar os polígonos.

O artigo apresenta um algoritmo com complexidade $O(nk^2 \log n)$ onde n é o número total de vértices dos polígonos e cercas, e k é o número de polígonos. Infelizmente, devido à natureza densa e teórica do artigo, não conseguimos implementar esse algoritmo com a complexidade prometida, mas conseguimos desenvolver uma implementação funcional que resolve o problema corretamente, embora com uma complexidade maior de $O(n^2k^2)$.

3.3 TPP com Polígonos Não Convexos

Nas duas variações anteriores do problema, consideramos que os polígonos eram convexos, isso é essencial para garintir que os problemas possam ser resolvidos em tempo polinomial, uma vez que podemos mostrar que até o TPP Irrestrito com polígonos não convexos é NP-difícil [1].

Ainda assim, decidimos explorar essa variação do problema, uma vez que ele não é descrito no artigo

original, implementando uma solução exata para o TPP Irrestrito com polígonos não convexos. A ideia geral do algoritmo é simples, basta decompor cada polígono não convexo em um conjunto de polígonos convexos, e então resolver o TPP Irrestrito normalmente, considerando que visitar o polígono não convexo significa visitar ao menos um dos polígonos convexos que o compõem. Dessa forma, reutilizamos a implementação do TPP Irrestrito que já havíamos desenvolvido.

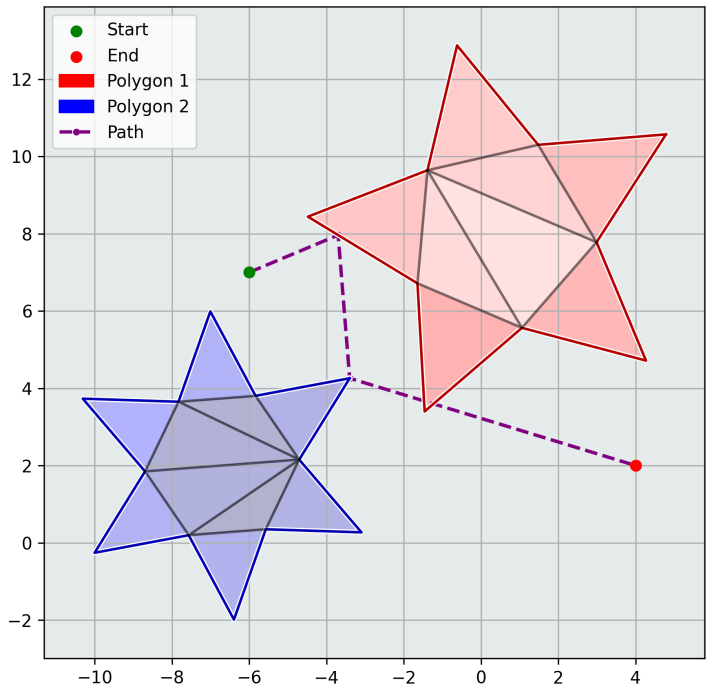

Figura 4: Caminho mínimo para um caso de 2 polígnos não convexos.

Utilizando essa abordagem, conseguimos resolver o problema corretamente, como ilustrado na figura ao lado 4, onde o caminho mínimo (linha roxa) visita os polígonos não convexos (polígonos preenchidos) corretamente, após considerar todas as combinações possíveis de polígonos convexos que os compõem.

No entanto, essa abordagem é extremamente ineficiente, uma vez que frequentemente vamos lidar com uma quantia exponencial de casos para testarmos. Enquanto não tivemos tempo de explorar otimizações para esse algoritmo, acreditamos que é possível usar técnicas de otimização inteira e combinatória para reduzir o número de casos que precisamos testar, tornando o algoritmo mais viável na prática. Esse tratamento seria feito no próximo semestre e muito provavelmente renderia um artigo científico.

Embora não tenhamos conseguido explorar essa variação muito profundamente ainda, a ideia central é tratar o problema como um problema deotimização inteira de uma função quadrática sujeita a restrições lineares. Dessa forma podemos usar técnicas de otimização matemática para otimizar a abordagem inicial, assim como utilizar solvers comerciais para comparar resultados e validar nossa implementação.

4 Resultados

Consideramos esse projeto como um sucesso, uma vez que conseguimos implementar soluções corretas para todas as variações do problema que propusemos, permitindo o aprendizado profundo sobre o problema e suas soluções. No entanto, na proposta inicial do projeto, fizemos diversas promessas amplas, assim, vamos discutir cada uma delas e analisar se conseguimos cumprir o que prometemos.

4.1 Mês 1

> Implementar a variação sem cercas, com estudo do artigo e testes iniciais. Estima-se cerca de 30 horas, divididos entre uma compreensão dos conceitos fundamentais e criação de uma base sólida de código para as etapas seguintes, assim como a criação de testes unitários para garantir a corretude do algoritmo.

Para o primeiro mês, prometemos que implementaríamos o TPP Irrestrito, estudando o artigo original e realizando testes iniciais. De fato, conseguimos cumprir essa promessa, implementando o algoritmo proposto no artigo [1] com a complexidade esperada, também realizamos diversos testes para garantir a corretude do algoritmo, criando uma base sólida de código que serviu como base para as variações seguintes do problema.

Embora tenhamos estimado que investiríamos cerca de 30 horas nesse mês, na prática, acabamos investindo muito mais tempo, cujos detalhes serão discutidos na seção de alocação de tempo. No entanto, acreditamos que esse investimento extra foi justificado, uma vez que nos permitiu compreender profundamente o problema e desenvolver uma implementação robusta e eficiente.

4.2 Mês 2

> Implementar a variação com cercas, utilizando parte do código da primeira variação. Esta etapa, estimada em 30 horas, envolve o aprofundamento do artigo original, adaptação e extensão do código para lidar com a complexidade adicional desta variação e criação de testes para validar a nova implementação.

Para o segundo mês, prometemos implementar o TPP Restrito, reutilizando parte do código desenvolvido para o TPP Irrestrito. De fato, conseguimos cumprir essa promessa, adaptando e estendendo nossa implementação anterior para lidar com as cercas do problema.

Infelizmente, devido à dificuldade do problema, não conseguimos atingir a complexidade prometida no artigo original [1], mas desenvolvemos uma implementação funcional que resolve o problema corretamente, embora com uma complexidade maior. Assim como na variação anterior, realizamos diversos testes para garantir a corretude do algoritmo.

Novamente, embora tenhamos estimado um investimento de 30 horas para esse mês, na prática, acabamos investindo muito mais tempo do que o previsto, levando muito tempo para compreender o artigo, implementar o algoritmo e lentamente depurar a implementação. Os detalhes desse investimento extra serão discutidos na seção de alocação de tempo.

4.3 Mês 3

> Explorar soluções para o caso de polígonos não convexos, mesmo que pouco eficientes. Prevê-se cerca de 30 horas, incluindo a idealização de uma abordagem, assim como a implementação dessa solução. Novamente, testes serão fundamentais para validar a solução proposta, mesmo que não seja a mais eficiente.

Para o terceiro mês, prometemos explorar soluções para o TPP Irrestrito com polígonos não convexos, mesmo que essas soluções fossem pouco eficientes. De fato, conseguimos cumprir essa promessa, desenvolvendo uma abordagem que decompõe os polígonos não convexos em polígonos convexos e reutiliza a implementação do TPP Irrestrito para encontrar o caminho mínimo.

Ao contrário dos meses anteriores, conseguimos cumprir a estimativa de 30 horas para esse mês, encontrando poucos contratempos durante a implementação e depuração do algoritmo. No entanto, reconhecemos que essa abordagem é extremamente ineficiente, e há muito espaço para melhorias e otimizações futuras.

4.4 Mês 4

> Desenvolver um software de visualização que permita ao usuário interagir com polígonos, cercas e caminhos encontrados, mostrando também as etapas do algoritmo. Estima-se 20 horas, incluindo criação de interface básica e implementação das funcionalidades de visualização.

Finalmente, para o último mês, prometemos desenvolver um software de visualização que permita ao usuário interagir com os polígonos, cercas e caminhos encontrados, mostrando também as etapas do algoritmo.

De fato, conseguimos cumprir essa promessa, desenvolvendo uma ferramenta de visualização que permite ao usuário carregar diferentes instâncias do problema, visualizar os polígonos e cercas, e observar o caminho mínimo encontrado pelo algoritmo. A ferramenta também permite visualizar as etapas do algoritmo, facilitando a compreensão do processo de resolução do problema. Por exemplo, todas as figuras apresentadas nesse relatório foram geradas utilizando essa ferramenta de visualização.

Além disso, adicionalmente, investimos tempo extra para melhorar a interface do usuário e adicionar um software extra de geração de instâncias do problema, facilitando a criação de novos casos de teste para o algoritmo, sem que o usuário tenha que digitar manualmente cada coordenada dos polígonos e cercas, podendo simplesmente desenhá-los e visualizar como isso afeta o caminho mínimo encontrado em tempo real.

O tempo investido nessa etapa também acabou sendo consideravelmente maior do que o estimado, uma vez que decidimos adicionar funcionalidades extras e melhorar a experiência do usuário. A implementação também acabou sendo mais desafiadora do que o previsto inicialmente, levando mais tempo para depurar e garantir que tudo funcionasse corretamente. Novamente, os detalhes desse investimento extra serão discutidos na seção de alocação de tempo.

5 Trabalhos Futuros

Enquanto consideramos que esse projeto foi muito bem sucedido, ainda há muito espaço para melhorias e trabalhos futuros. Algumas das possíveis direções para trabalhos futuros incluem:

- Otimização do algoritmo para o TPP Restrito, buscando atingir a complexidade prometida no artigo original [1], ou pelo menos melhorar a complexidade atual da implementação.
- Exploração de otimizações para o TPP Irrestrito com polígonos não convexos, utilizando técnicas de otimização inteira e combinatória para reduzir o número de casos que precisam ser testados. Essa tarefa provavelmente será um tanto quanto complicada e seria tratada ao longo do próximo semestre, possivelmente rendendo um artigo científico.
- Melhorias na ferramenta de visualização, adicionando mais funcionalidades e melhorando a experiência do usuário.

Quaisquer avanços podem ser acompanhados no mesmo repositório que estamos usando atualmente, onde também é possível encontrar o código desenvolvido até agora, assim como instruções para executar os algoritmos implementados.

É extremamente provável que esses trabalhos futuros sejam explorados no próximo semestre, como parte de uma continuação desse projeto de pesquisa, e que a participação nessa matéria seja novamente registrada como parte da atividade curricular em pesquisa.

6 Alocação de Tempo

Como requisito para a aprovação na disciplina MAC0215 - Atividade Curricular em Pesquisa, é necessário comprovar um investimento de ao menos 100 horas de trabalho no projeto. A seguir, vamos detalhar como esse tempo foi alocado ao longo do semestre, justificando o investimento realizado.

Primeiramente, seria possível que simplesmente disséssemos que uma certa quantia de horas foi investida em cada mês, mas essa métrica é muito vaga e não reflete o esforço real investido no projeto. Por esse motivo, desenvolvemos uma métrica objetiva para medir o tempo investido, baseada nos múltiplos *commits* realizados no repositório ao longo do projeto.

6.1 Uma Heurística Para Medição de Tempo

Essencialmente, cada *commit* representa uma unidade de trabalho realizada no projeto, seja implementando uma nova funcionalidade, corrigindo um bug ou escrevendo documentação. No entanto, nem todos os *commits* são iguais, alguns representam um esforço maior do que outros, podendo variar desde pequenas correções feitas em poucos minutos até implementações complexas que levaram múltiplas horas para serem concluídas.

Para não fazermos decisões arbitrárias sobre o esforço investido em cada *commit*, decidimos aproveitar que cada *commit* registra o tempo exato em que foi realizado, permitindo que calculemos o tempo entre cada *commit* consecutivo. Dessa forma, podemos somar o tempo entre todos os *commits* realizados ao longo do projeto, obtendo uma métrica objetiva do tempo investido.

No entanto, é claro que essa métrica não é perfeita, uma vez que há momentos em que o trabalho no projeto é interrompido por longos períodos, como pausas para refeições, descanso ou outras atividades. Para mitigar esse problema, decidimos estabelecer um limite máximo de 3 horas entre *commits* consecutivos, ou seja, se o tempo entre dois *commits* for maior do que 3 horas, desconsideramos esse intervalo na soma total, assumindo que o trabalho foi interrompido nesse período.

Essa abordagem é um tanto conservadora, uma vez que sempre ignora o tempo investido no primeiro *commit* do dia, assim como qualquer intervalo maior do que 3 horas, mas acreditamos que é uma forma justa de medir o tempo investido no projeto, evitando inflar artificialmente a métrica com períodos de inatividade.

6.2 Tempo Investido em Commits

Uma vez que definimos nossa métrica para medir o tempo investido no projeto, podemos aplicar essa métrica aos mais de 200 *commits* realizados ao longo do semestre, obtendo um total de aproximadamente 100 horas de trabalho investidas no projeto, o que excede o requisito mínimo de 100 horas para aprovação na disciplina.

Ao lado, criamos um gráfico 5 que ilustra o tempo investido ao longo do projeto, medido em horas ao longo do tempo. Podemos observar que o tempo investido varia ao longo do semestre, com picos de atividade em determinados períodos, refletindo o esforço concentrado em diferentes etapas do projeto.

Esses picos de atividade geralmente coincidem com as etapas principais do projeto, assim como um período onde tivemos mais tempo livre para trabalhar no projeto, como durante a Semana Santa ou Semana de Break da computação. No entanto, também há períodos de menor atividade, refletindo períodos de provas, outras responsabilidades acadêmicas, pessoais ou simplesmente um período de descanso que limitaram o tempo disponível para o projeto.

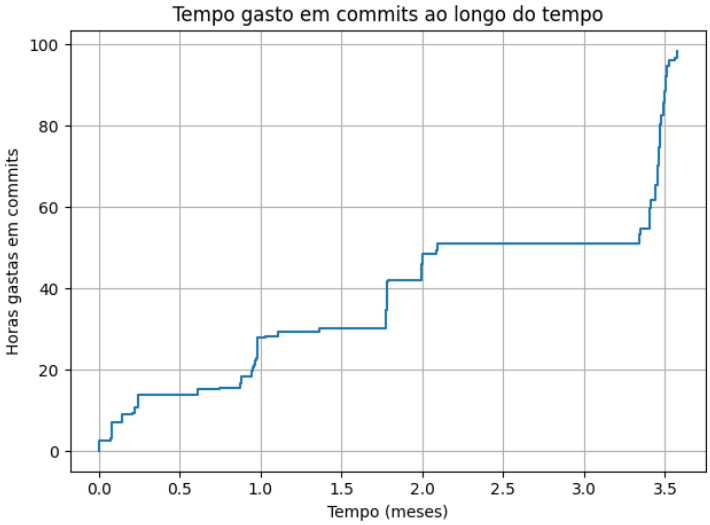

Figura 5: Tempo investido ao longo do projeto, medido em horas ao longo do tempo.

Além disso, um fator perceptível no gráfico é um período longo de mais de um mês sem nenhum *commit*, que ocorreu entre o final de setembro e o início de novembro. O motivo para esse hiato é que no início do semestre submetemos uma proposta de projeto de pesquisa para a Fapesp a respeito desse trabalho, no entanto, no final de setembro, recebemos a notícia de que a proposta não foi aprovada, o que causou tanto uma desmotivação temporária quanto uma necessidade de reescrita da proposta e planejamento do projeto, o que acabou atrasando o progresso do projeto em cerca de um mês.

Em compensação, notamos um grande pico de atividade logo após esse hiato, refletindo o esforço concentrado para recuperar o tempo perdido e avançar no projeto. Apesar desse contratempo, conseguimos retomar o ritmo do projeto e avançar significativamente nas etapas seguintes, conforme ilustrado no gráfico.

6.3 Outros Fatores Considerados

Na seção anterior, já observamos que mais de 100 horas foram investidas exclusivamente em *commits* no repositório, mas é claro que esse não é o único fator que deve ser considerado na alocação de tempo do projeto. Além do tempo investido em *commits*, também devemos considerar o tempo investido em outras atividades relacionadas ao projeto, como o estudo do tema, leitura de artigos, planejamento do projeto, reuniões com o orientador e outras atividades que contribuíram para o progresso do projeto.

Dentre essas atividades, o estudo do tema e leitura de artigos foram particularmente importantes, uma vez que o problema abordado é complexo e requer um entendimento profundo dos conceitos envolvidos. Investimos tempo significativo na leitura do artigo original [1], uma vez que ele é denso e teórico, exigindo uma análise cuidadosa para compreender as ideias apresentadas e como implementá-las na prática. Também muito tempo foi investido na busca por artigos relacionados, uma tarefa que se mostrou mais difícil do que o esperado, uma vez que há poucos artigos que tratam diretamente do TPP, e a maioria das soluções propostas ainda se baseia nas ideias apresentadas no artigo original.

Além do estudo do tema, também investimos tempo em diversas reuniões com o orientador, Prof. Dr. Ernesto G. Birgin, que foram fundamentais para o progresso do projeto. Nessas reuniões, discutimos o andamento do projeto, esclarecemos dúvidas, recebemos feedback e orientações, e planejamos as próximas etapas do trabalho. Essas reuniões foram essenciais para garantir que o projeto estivesse no caminho certo e para superar os desafios encontrados ao longo do caminho. As reuniões ocorreram aproximadamente a cada duas semanas, com duração média de duas horas, totalizando cerca de 16 horas investidas em reuniões ao longo do semestre.

Outro fator importante a ser considerado é o tempo investido nessa matéria, com a escrita do relatório final e vídeo de apresentação. Como o leitor pode perceber, o relatório final é bastante detalhado e extenso, exigindo um esforço significativo para redigir, revisar, formatar o documento e gerar as figuras apresentadas.

Também podemos indicar uma quantia significativa de tempo investida na confecção da proposta para a Fapesp, consumindo diversas horas de planejamento, redação e revisão do documento, além de reuniões com o orientador para discutir e aprimorar a proposta e o tempo para conseguir os documentos necessários para

a submissão. Além disso, esse processo todo foi repetido para a reconsideração. É claro que esse tempo não contribuiu diretamente para o progresso do projeto, mas foi uma parte importante do processo de pesquisa e desenvolvimento do trabalho.

Um último fator relevante a ser considerado é a participação no SIICUSP, um evento da USP onde apresentamos o projeto para tanto alunos quanto professores do instituto. Esse evento exigiu tempo significativo de preparação, incluindo a criação de um pôster, slides, preparação de uma apresentação oral e prática para garantir que estivéssemos prontos para apresentar o trabalho de forma clara e concisa, além das horas ativamente apresentando o trabalho para os interesados. A participação no SIICUSP foi uma oportunidade valiosa para compartilhar nosso trabalho com a comunidade acadêmica, receber feedback e estabelecer conexões com outros pesquisadores interessados no tema, embora tenha sido um tanto cansativa.

Enquanto não conseguimos quantificar exatamente o tempo investido nessas atividades adicionais, acreditamos que o tempo total investido no projeto, considerando tanto os *commits* quanto essas outras atividades, excede confortavelmente o requisito mínimo de 100 horas para aprovação na disciplina MAC0215 - Atividade Curricular em Pesquisa.

7 Conclusão

Nesse relatório, apresentamos o trabalho realizado ao longo do semestre como parte da disciplina MAC0215 - Atividade Curricular em Pesquisa, onde exploramos o Problema da Visita de Polígonos (TPP) e suas variações. Discutimos o problema em si, a base bibliográfica que fundamenta nosso trabalho, detalhamos as implementações realizadas, analisamos os resultados obtidos e propusemos possíveis melhorias e trabalhos futuros.

A respeito dos resultados, consideramos o projeto como um sucesso, uma vez que conseguimos implementar soluções corretas para todas as variações do problema que propusemos, permitindo o aprendizado profundo sobre o problema e suas soluções. No entanto, reconhecemos que há espaço para melhorias e trabalhos futuros, que pretendemos explorar no próximo semestre.

Além disso, detalhamos a alocação de tempo ao longo do semestre, justificando o investimento de muito mais de 100 horas de trabalho no projeto, cumprindo assim o requisito para aprovação na disciplina.

Agradecemos ao Prof. Dr. Ernesto G. Birgin pela orientação e suporte ao longo do projeto, assim como à comunidade acadêmica que contribuiu para o desenvolvimento desse trabalho.

Finalmente, encorajamos o leitor a explorar o código desenvolvido para esse projeto, disponível no repositório do GitHub, assim como acompanhar desenvolvimentos futuros relacionados a esse trabalho.

Referências

[1] M. Dror, A. Efrat, A. Lubiw e J. S. B. Mitchell, "Touring a sequence of polygons," em *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, sér. STOC '03, San Diego, CA, USA: Association for Computing Machinery, 2003, pp. 473–482, ISBN: 1581136749. DOI: 10.1145/780542.780612. endereço: https://doi.org/10.1145/780542.780612.

[2] I. Gentilini, F. Margot e K. Shimada, "The travelling salesman problem with neighbourhoods: MINLP solution," *Optimization Methods and Software*, v. 28, n. 2, pp. 364–378, 2012. endereço: https://doi.org/10.1080/10556788.2011.648932.

[3] D. Applegate, R. Bixby, V. Chvátal e W. Cook, "The Traveling Salesman Problem: A Computational Study," *The Traveling Salesman Problem: A Computational Study*, jan. de 2006.