BRUNO FREITAS RÊGO
GABRIEL ROCHA AMORIM
GUILHERME YUITI MIAZAQUI

GERENCIAMENTO DE DESASTRES: UMA ABORDAGEM BASEADA EM SISTEMAS MULTIAGENTES

São Paulo
2020

BRUNO FREITAS RÊGO
GABRIEL ROCHA AMORIM
GUILHERME YUITI MIAZAQUI

GERENCIAMENTO DE DESASTRES: UMA ABORDAGEM BASEADA EM SISTEMAS MULTIAGENTES

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

São Paulo
2020

BRUNO FREITAS RÊGO
GABRIEL ROCHA AMORIM
GUILHERME YUITI MIAZAQUI

GERENCIAMENTO DE DESASTRES: UMA ABORDAGEM BASEADA EM SISTEMAS MULTIAGENTES

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

Área de Concentração:
Sistemas Multiagentes

Orientador:
Prof. Dr. Jaime Simão Sichman

Co-orientador:
Prof. Dr. Luis Gustavo Nardin

São Paulo
2020

Catalogação-na-publicação

Freitas Rêgo, Bruno
GERENCIAMENTO DE DESASTRES: UMA ABORDAGEM BASEADA EM SISTEMAS MULTIAGENTES / B. Freitas Rêgo, G. Rocha Amorim, G. Yuiti Miazaqui -- São Paulo, 2020.
73 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Inteligência Artificial 2.Times de Agentes 3.RoboCup Rescue Simulation 4.Aprendizado não supervisionado 5.K-Means I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t. III.Rocha Amorim, Gabriel IV.Yuiti Miazaqui, Guilherme

AGRADECIMENTOS

Ao Prof. Dr. Jaime Simão Sichman e ao Prof. Dr. Luis Gustavo Nardin que nos orientaram durante todo o desenvolvimento deste trabalho, oferecendo apoio, conhecimento, sugestões, críticas e dedicação para nos auxiliar na elaboração deste projeto. Em especial, às noites de depuração de código e madrugadas no caso de Gustavo por conta do fuso. Pela paciência, e constante apoio, nosso muito obrigado.

Ao desenvolvedor iraniano Mahdi Taherian que atuou na criação do Visual Debugger utilizado na RoboCup Rescue Simulation para estender as capacidades de compreensão do comportamento das entidades da simulação e que nos apoiou na adaptação da ferramenta às necessidades deste trabalho.

Aos nossos pais, por valorizarem a educação e nos permitirem priorizar os estudos e que foram essenciais nessa jornada de graduação.

À Escola Politécnica da USP e todos seus docentes e colaboradores que nos propiciaram uma excelente formação técnica e aprendizados para toda a vida.

Por fim, a todos que agregaram conhecimentos durante este período, desde experiências extra curriculares à experiências profissionais, agradecemos todos que fizeram parte dessa jornada.

RESUMO

Sistemas Multiagentes (SMA) têm ganhado relevância na resolução de problemas reais e complexos, e exemplo disso é a competição RoboCup Rescue. Simula-se um desastre natural com sistemas multiagentes e busca-se criar algoritmos capazes de minimizar ao máximo os impactos do desastre.

Uma das abordagens possíveis e abordadas neste trabalho é o particionamento da região de exploração utilizando o algoritmo de clusterização K-Means. Com a motivação de maximizar a quantidade de civis resgatados e melhorar a eficiência dos agentes, a arquitetura proposta implementa a clusterização centralizada no início da simulação considerando a área dos prédios de cada região. Além disso, foi proposta a centralização de comandos em oposição ao comportamento autônomo padrão dos agentes. Essa proposta foi implementada em dois níveis diferentes, em que no primeiro somente a alocação de alvos seria feita, enquanto no segundo a exploração de edifícios seria também comandada além da atribuição dos alvos.

Para avaliação de resultados, foram realizadas 450 coletas de dados distribuídas em cinco diferentes mapas da simulação, buscando mensurar os impactos proporcionados pelas estratégias propostas. Enquanto a métrica principal utilizada foi a pontuação final dada pelo próprio simulador, outras foram utilizadas como número de civis resgatados e número de civis encontrados. Concluiu-se que, apesar de ter sido encontrada diferença estatisticamente relevante na pontuação final em um dos mapas para K's diferentes, o ganho advindo da organização de ações de forma centralizada foi superior.

Palavras-Chave – clusterização, RoboCup Rescue Simulation, K-Means.

ABSTRACT

Multi-agent systems (SMA) have gained relevance in solving real and complex problems, and an example of this is the RoboCup Rescue competition. A natural disaster is simulated with a multi-agent system and attempts are made to create algorithms capable of minimizing the impact of the accident.

One of the possible approaches that is addressed in this work is the partitioning of the exploration region using the K-Means clustering algorithm. With the motivation of maximizing the quantity of civilians saved and improving agents' efficiency, the proposed architecture implements centralized clustering at the beginning of the simulation considering the area of the buildings in each region. Furthermore, centralizing commands was proposed in oposition to standard autonomous behavior of agents. This proposal was implemented in two different levels, where in the first only target allocation would be made, while in the second exploring the buildings would also be commanded in addition to the target allocation.

To evaluate the results, 450 data collections were carried out spread across five different simulation maps, seeking to measure the impacts provided by the proposed strategies. While the main utilized metric was the final score provided by the simulator itself, others were used such as the number of rescued civilians and the number of identified civilians. It was concluded that, although a statistically significant difference was found in the final score in one of the maps for different K's, the gain achieved from organizing actions in a centralized manner was higher.

Keywords – clustering, RoboCup Rescue Simulation, K-Means.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

"Desastres naturais são eventos adversos e impactantes que causam perdas econômicas, humanas e ambientais em larga escala. Eles geralmente são difíceis de prever e é ainda mais desafiador impedi-los de acontecer. Essas características exigem estratégias de gerenciamento de desastres para mitigar consequências prejudiciais quando um desastre acontece."

-- RoboCup Rescue Simulation League

Na história da Inteligência Artificial, também conhecida por IA, o ano de 1997 foi marcado pela derrota do até então campeão mundial de xadrez Gary Kasparov pelo computador Deep Blue, construído pela IBM [1]. Esse fato mostrava que uma nova era na IA iniciava e, no mesmo ano, ocorreu a primeira competição e conferência oficial do RoboCup, um veículo para promover o estudo da robótica e estudos relacionados à IA. Mais de 40 times participaram e 5.000 espectadores partiparam do evento e tratou-se apenas de um primeiro passo do ambicioso objetivo de longo prazo: criar um time de humanóides completamente autônomo que seja capaz de vencer o último ganhador da Copa do Mundo até o meio do século XXI. [2]

A complexidade na criação de um humanoide com tamanha capacidade técnica requer amplo domínio em diferentes áreas do conhecimento da IA. Para isso, foram criadas as seguintes ligas: RoboCup Soccer, RoboCup Rescue, RoboCup Home, Robocup Industrial e a RoboCup Junior. Cada uma delas atua buscando gerar um impacto social positivo por meio de aplicações em problemas reais, a exemplo da RoboCup Industrial que utiliza conceitos de outras competições para aplicação em robôs industriais. Este trabalho possui enfoque na RoboCup Rescue, que foi criada em 2001 como resposta ao Grande Sismo de Hanshin-Awaji que atingiu o Japão em 17 de janeiro de 1995. Simula-se um desastre natural com sistemas multiagentes, tema que tem ganhado um renovado interesse nos últimos anos, em parte pelo aumento de desempenho dos computadores e a disponibilidade destes em ambientes de nuvem. Além disso, possuem ampla aplicação na resolução de problemas reais e complexos, como planejamento de veículos autônomos [3], aplicações em ambientes industriais [4], transferência de conhecimento [5] e gestão de desastres naturais [6].

Quando se trata da gestão de desastres naturais, incertezas e dificuldades são inerentes ao processo, dado que o conhecimento do estado atual do ambiente é muitas vezes incerto ou parcial. Essas características dificultam a alocação de tarefas relacionadas ao resgate, e impossibilitam a utilização de estratégias que requerem conhecimento prévio da situação do desastre. Uma possibilidade para superar esse problema é utilizar algoritmos de exploração de ambiente baseados em particionamento, também conhecido como clusterização, foco deste trabalho.

1.1 OBJETIVO

O objetivo deste trabalho é analisar o impacto da clusterização do ambiente na pontuação final e no desempenho de resgate dos civis. Também é um objetivo melhorar a colaboração de agentes autônomos no planejamento e na execução de resgates através da centralização de comandos. Para isso, propõe-se desenvolver uma equipe de resgate que utiliza a clusterização na distribuição inicial dos agentes pelo ambiente por meio da utilização do algoritmo K-Means [7].

A setorização é realizada já no início da simulação, segregando áreas similares em um mesmo cluster utilizando a distância Euclidiana e a concentração da prédios em cada setor, utilizando nessa avaliação a área total de cada edifício. A criação do algoritmo de clusterização busca ganhos tangíveis na simulação, tais como a redução do tempo médio para resgate de vítimas e o aumento do número de sobreviventes, por exemplo, fatos decorrentes da melhor alocação de recursos humanos. A avaliação destes resultados obtidos será feita por meio de comparações estatísticas que buscam avaliar a eficácia do algoritmo desenvolvido, utilizando testes como o Mann-Whitney-Wilcoxon, um teste não paramétrico que é aplicado para duas amostras independentes de resultados. O algoritmo também é avaliado em diferentes mapas da simulação, buscando minimizar vieses geográficos que podem existir por conta de particularidades de certos mapas, tais como a maior concentração de civis em áreas litorâneas do mapa ou regiões com baixa densidade de prédios.

1.2 MOTIVAÇÃO

Formando uma sub-área da Inteligência Artificial, os Sistemas Multiagentes são compostos por agentes independentes que interagem entre si com um conjunto de ações possíveis. Estas interações ocorrem em um ambiente determinado e são regidas por uma

organização ou por uma política de conduta. Já cada um destes agentes [8] é um sistema computacional situado em um ambiente em que pode agir autonomamente para atingir seus objetivos.

É grande o potencial de resolução de problemas de tais sistemas, e a quantidade de formas diferentes de criá-los também não fica para trás. Observa-se integração de várias áreas da computação para cada aplicação, sendo que ao se aplicar, por exemplo, Aprendizado de Máquina, vê-se Aprendizagem Profunda, Aprendizado por Reforço e Redes Neurais como algumas das formas de aplicação. Também, em modelos de aprendizado não supervisionado, pode-se citar K-Means, Modelos Ocultos de Markov e Clusterização Hierárquica como exemplos [9].

Através do Aprendizado de Reforço [10], observam-se ações emergentes motivadas pelas recompensas. Surgem comportamentos inusitados e inteligentes dentro de um ambiente complexo, com cada um dos agentes objetivando superar novas barreiras. Estas estratégias emergentes são evidências de que soluções inusitadas podem surgir destes sistemas, e, no futuro, tarefas complexas do mundo real podem ser melhoradas através da constatação da maneira que estes agentes as resolvem.

É com esta motivação que a RoboCup League organiza torneios de simulação de resgate em desastres. A cada ano, novos times tentam melhorar o algoritmo de seus agentes para obter uma pontuação mais alta e, ao mesmo tempo, contribuem um pouco para um sistema cada vez melhor de gerenciamento de desastres. Isso, entre outras razões, acontece por conta da escassez de dados reais de desastres, o que refreia o aprendizado com experiências passadas. A simulação pode, neste sentido, servir como um guia de decisões para um desastre real, embora seja um modelo que deve ser seguido com certa cautela, dado à simplificação característica de tais simulações.

Adicionalmente, a RoboCup League Simulation está inserida no contexto do objetivo de longo prazo proposto pela RoboCup e já apresentado na introdução deste trabalho: criar um time de humanóides completamente autônomo que seja capaz de vencer o último ganhador da Copa do Mundo até o meio do século XXI [2]. Dessa forma, o aprimoramento da simulação de desastres naturais, que envolvem incertezas e inúmeras variáveis, contribui indiretamente para o desenvolvimento do humanóide que invariavelmente lida com situações inesperadas e instáveis que também estão presentes em uma partida de futebol.

1.3 JUSTIFICATIVA

A setorização do ambiente para ação dos agentes exploradores se mostrou eficaz em alguns cenários quando comparada a um modelo sem setorização, o que é mostrado em [11]. Dentre as conclusões, também é observado que o método é uma solução promissora para a melhora de desempenho dos sistemas. Neste contexto, pretende-se ao longo deste trabalho explorar ainda mais o potencial da setorização, desta vez com a Central de Ambulâncias realizando a clusterização no início da simulação e considerando a relevância da área dos prédios nessa divisão: quanto mais prédios em uma região, maior a necessidade de entidades para exploração da área.

Tem crescido nos últimos anos empresas que aplicam soluções de Inteligência Artificial como resposta a desastres naturais. Em 2015, dois terremotos podem ser citados nos quais tecnologia foi empregada para mitigar seus efeitos, no Nepal e no Chile [12]. No primeiro país, *tweets* e fotos foram categorizados por voluntários para que servissem de entrada em um sistema baseado em IA, que categorizou urgências, danos e recursos necessários com base nestes dados. Já no Chile, simulações de desastres passaram a ser feitas anualmente, e a eficiência com que o país conseguiu realizar a evacuação deve méritos a esta iniciativa. O sistema de comunicação criado alertou todos os cidadãos na área do terremoto para saírem da região, evitando mortes por tsunamis.

O sucesso destas operações mostra que há serventia em simulações, e também que tecnologia contribui muito para respostas mais efetivas aos desastres. É neste contexto de geração de impactos positivos à sociedade por meio da utilização de técnicas de Inteligência Artificial que este trabalho se insere.

1.4 ORGANIZAÇÃO DO TRABALHO

A seguir está descrita a organização do restante da monografia, mostrando o principal objetivo de cada um dos oito capítulos:

No capítulo 1 deste trabalho, foram definidos o problema, o contexto em que ele está inserido, a motivação da realização deste sistema e o que buscamos alcançar neste projeto.

No capítulo 2, realizamos a contextualização dos conceitos de suporte para a compreensão do trabalho e a revisão da literatura de base. Os conceitos são apresentados fornecendo o nível de complexidade necessário para o entendimento pleno dos capítulos posteriores.

No capítulo 3, listamos as ferramentas, algoritmos e dados necessários para o desenvolvimento da clusterização.

No capítulo 4, definimos os processos e fases no desenvolvimento de funcionalidades da clusterização.

No capítulo 5, definimos os requisitos técnicos para o desenvolvimento da clusterização, apresentando restrições que devem ser respeitadas e necessidades funcionais na simulação.

No capítulo 6, definimos os processos de implementação das tecnologias e requisitos levantados nos capítulos anteriores, apresentando problemas que foram encontrados e suas respectivas soluções.

No capítulo 7, estão documentados os resultados do sistema através de testes feitos considerando as métricas apresentadas no capítulo 4.

No capítulo 8, relatamos a experiência de projeto do Trabalho de Conclusão de Curso e apresentamos abordagens adicionais que restaram para uma avaliação futura, incluindo também a avaliação do cumprimento dos objetivos propostos no capítulo 1 deste trabalho e a conclusão final da monografia.

2 ASPECTOS CONCEITUAIS

Com as informações iniciais do projeto apresentadas no capítulo anterior, iremos nos aprofundar na base teórica necessária para entendimento e posterior implementação do algoritmo K-Means de clusterização. Dentre estes aspectos, discorreremos neste capítulo sobre os conceitos do contexto em que o problema está inserido, apresentando o simulador e o projeto Robocup Rescue, algoritmos de classificação e definições da avaliação estatística utilizadas no capítulo 7.

2.1 CONTEXTO

A Inteligência Artificial Distribuída (IAD) é uma área da Inteligência Artificial (IA) que nasceu nos anos 80, com o objetivo de trazer uma abordagem focada na resolução de problemas complexos, baseada na distribuição de tarefas e capacidades e também na comunicação entre agentes. Buscando semelhança na forma que muitos problemas são resolvidos em grupos de pessoas, sugere a criação de vários agentes inteligentes que cooperam para a resolução, em contraste com a abordagem de um único agente complexo [13].

As propriedades da IAD, entretanto, podem ser implementadas de maneiras diferentes e com focos distintos. Neste sentido ainda, também por motivações históricas e científicas, a forma com que a arquitetura destas soluções é implementada caracteriza uma de duas subáreas diferentes da IAD. A primeira é a Resolução Distribuída de Problemas (RDP), e a segunda Sistemas Multiagentes (SMA).

A RDP considera que a resolução de problemas pode ser dividida em diferentes módulos ou nós que cooperam entre si por meio da divisão de tarefas e compartilhamento de conhecimento sobre o problema. Nessa abordagem, a concepção dos agentes, de sua organização e de suas interações é consequência da existência de um grande problema específico que busca ser solucionado e que motiva a criação do sistema. Dessa maneira, a reutilização desses agentes, de seus modelos organizacionais e de suas integrações é pouco

provável, ou seja, de certo modo pode-se considerar a RDP como uma mescla de técnicas de IA e de sistemas distribuídos [14].

A segunda área, a de SMA, aborda a coordenação do comportamento, conhecimento, objetivos, habilidades e planos dos agentes para juntos tomarem ações e resolver problemas. Em Sistemas Multiagentes a tarefa de coordenação pode ser difícil, já que os agentes possuem uma existência própria e a concepção deles, assim como de seu sistema e modelos organizacionais, não são necessariamente consequência de um problema particular a ser solucionado. Torna-se possível, portanto, a reutilização desses agentes em outras aplicações [15].

Assim, um agente é uma entidade computacional ou real que é capaz de agir em um ambiente e é movida por objetivos individuais ou comuns e metas, e que pode se comunicar diretamente ou indiretamente com outras. Seu comportamento é consequência de percepções, representações e interações com o ambiente ou com outros agentes. Quando em conjunto, formam os chamados Sistemas Multiagentes [16].

2.2 SISTEMAS MULTIAGENTES

Na perspectiva de um agente individual, os Sistemas Multiagentes diferem de sistemas de um único agente principalmente nas dinâmicas do ambiente que podem ser determinadas por outros agentes. Outros agentes podem afetar o ambiente de maneira imprevisível, adicionando incertezas que remodelam o ambiente. Essa imprevisibilidade se explica no fato de que, como cada um destes existe por si só, possuem, consequentemente, conhecimentos diferentes entre si. Podem existir, ainda, com diferentes graus de heterogeneidade, tendo ou não habilidade de comunicação direta com outros agentes.

Um exemplo de domínio que requer o uso de um SMA e não pode ser contemplado pela RDP para representar os objetivos de cada agente é um caso de estudo de um hospital. Cada funcionário do hospital tem diferentes interesses e metas: enfermeiras buscam minimizar o tempo que o paciente passa no hospital, enquanto operadores de máquinas de Raio-X buscam maximizar a quantidade de exames realizados, por exemplo.

A simulação do RoboCup Rescue se encaixa nesta categoria porque seus agentes possuem diferentes funções e conhecem informações diferentes, com cada uma destas funções objetivando aspectos diferentes do resgate. Esta taxonomia é abordada com mais detalhes na seção 3.1.1. Apesar disso, existe também um viés de RDP, já que existe uma hierarquia para distribuição de tarefas, e não são, portanto, totalmente livres para escolher as suas

próprias.

2.2.1 HETEROGENEIDADE

Sistemas Multiagentes podem ser homogêneos ou heterogêneos. Os homogêneos apresentam agentes idênticos, incluindo conhecimentos e ações possíveis. Apesar de possuírem as mesmas capacidades, possuem informações limitadas a respeito do estado de outros e das manipulações destes no ambiente, de forma que não conseguem prever as ações de outros. Apesar de possuírem uma mesma estrutura, os agentes recebem diferentes estímulos do ambiente, condição necessária para Sistemas Multiagentes. Caso contrário, todos agiriam da mesma forma, formando em suma uma única unidade [17].

Os sistemas heterogêneos possuem agentes com diferentes objetivos, ações, conhecimentos ou modelos. Neste sentido, podem ter ações e tratamentos complementares para solucionar um determinado problema. Apesar de possuírem diferentes objetivos, podem ser amigáveis com os objetivos de outros agentes. Isso resulta em um maior potencial de análise do sistema, apesar de aumentar sua complexidade [18].

2.2.2 DINAMICIDADE

Segundo Sichman e Garcia (2005) [14], com respeito à interação social entre agentes, existem duas classes de modelos, cujo desenvolvimento se deu por padrões de interações repetidos observados no ambiente. As duas categorias são Modelos Estáticos ou descendentes e Modelos Dinâmicos ou ascendentes.

A primeira aborda agentes que já possuem objetivo em comum desde sua concepção. Neste caso a cooperação entre eles é tida como certa e tomada como hipótese de partida. As interações entre agentes ficam limitadas por uma organização pré-existente, como no caso da hierarquia que se propõe aqui implementar. Tarefas são distribuídas de cima pra baixo, e os agentes resgatadores de uma determinada função não podem se comunicar com centrais de outro papel (i.e policiais com a central de bombeiros).

Já nos modelos dinâmicos, também chamados de coalizão ou coligação, não existe um objetivo em comum, e as comunicações são pouco ou não regulamentadas. Por esse prisma, elas surgem como uma negociação de objetivos entre agentes. No contexto deste projeto, isso se concretiza na interação entre agentes centrais, uma vez que, em primeiro lugar, o mapa não começa já com um desastre em andamento conhecido e, em segundo lugar, que têm funções complementares entre si. Um exemplo disso então, é que o objetivo

dos policiais é descongestionar as vias e o dos bombeiros é apagar incêndios, duas metas diferentes porém complementares. Isso caracteriza um modelo baseado na complementaridade, em que a existência de outros agentes aumenta a autonomia e poder de cada um [14].

2.3 ALGORITMOS DE CLASSIFICAÇÃO

Existem diferentes abordagens computacionais que podem ser utilizadas para aprendizagem de máquina, dividindo-se essencialmente em três grandes grupos:

- Aprendizagem supervisionada: forma de aprendizagem mais comum que utiliza um conjunto de exemplos previamente classificados, a partir dos quais se consegue construir um modelo para posterior aplicação em exemplos novos. Chama-se de aprendizagem supervisionada por existir uma entidade responsável por fornecer exemplos adequados à aprendizagem;
- Aprendizagem não supervisionada: o algoritmo precisa obter as informações para a sua aprendizagem a partir das entradas de dados disponíveis. Muitos dos algoritmos nessa aprendizagem utilizam partições (clusters), classificando os dados com base em padrões de similaridade entre as entradas, sem a presença de um supervisor;
- Aprendizagem por reforço: utiliza as mesmas bases da aprendizagem supervisionada, porém sem a disponibilidade de informações como a saída desejada. Tem-se apenas a informações de se a saída apresentada pelo algoritmo está correta, e isso é utilizado para aprimorar os parâmetros do algoritmo para aprender os exemplos corretamente.

A clusterização se refere à ampla variedade de técnicas de definição de subgrupos dentro de um conjunto de dados, também chamados de clusters. As técnicas buscam particionar os dados em grupos distintos um dos outros, enquanto dados pertencentes ao mesmo grupo possuem similaridades. Neste trabalho é apresentado e utilizado o K-Means, um algoritmo de clusterização com aprendizagem não supervisionada.

O algoritmo K-Means tem como objetivo dividir um conjunto de dados em um determinado número k de grupos distintos (clusters), em que cada dado pertence ao cluster com a média mais próxima do próprio dado. O algoritmo é um dos mais utilizados na aprendizagem computacional pelo fato de ser simples e servir muitas vezes como um passo de pré-processamento para a aplicação de outros algoritmos. [19]

Dessa maneira, a divisão agrupa dados semelhantes dentro do mesmo cluster, sendo um algoritmo de aprendizagem não supervisionada, no qual a aprendizagem ocorre sem um conjunto de exemplos previamente classificados e sem a possibilidade de construção de um modelo prévio para posterior aplicação dos dados na classificação. O processo genérico de funcionamento do algoritmo parte de um conjunto de dados e de um número de clusters pré-definidos K, sendo composto pelos seguintes passos:

1. Definir um valor de K, ou seja, o número de clusters ou agrupamentos desejado.
2. Para cada um dos K clusters, definir aleatoriamente um centróide.
3. Calcular, para cada ponto, o centróide de menor distância (usualmente pela distância Euclidiana), de forma que cada ponto pertencerá ao centróide mais próximo.
4. Reposicionar o centróide, atribuindo a média da posição de todos os pontos de cada cluster como a nova posição de cada centróide.
5. Iterar pelos passos 3. e 4. até a convergência (minimização da soma dos desvios quadráticos do padrões dos clusters em relação aos seus centros), obtendo a posição ideal dos centroides.

O centróide representa a média de todos os dados do cluster, podendo ser um ponto que representa as características do conjunto de dados pertencente a um cluster e não necessariamente um dado real do conjunto de dados do problema. Visualmente, a divisão gerada pelo K-Means pode ser representada em um Diagrama de Voronoi, apresentando os clusters e seus respectivos centroides.

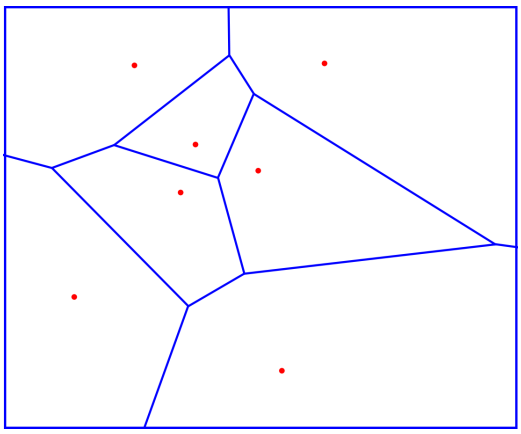


Figura 1: Diagrama de Voronoi da divisão do K-Means.

O progresso da execução da aplicação do K-Means para um conjunto de dados iniciais pode ser avaliado no exemplo apresentado em [20], adotando um valor de K como 3. No

canto superior esquerdo da figura, é apresentado o conjunto inicial de dados no qual data uma das amostras é atribuída aleatoriamente a um cluster (passo 1) e posteriormente calculam-se os 3 centroides. Inicialmente eles são mostrandos como discos praticamente sobrepostos pelo fato de terem sido aleatoriamente definidos (passo 2). Em seguida cada uma das amostras é reatribuída ao cluster mais próximo (passo 3) e posteriormente são reposicionados os centroides (passo 4), até a chegada ao resultado final da clusterização após 10 iterações (passo 5).

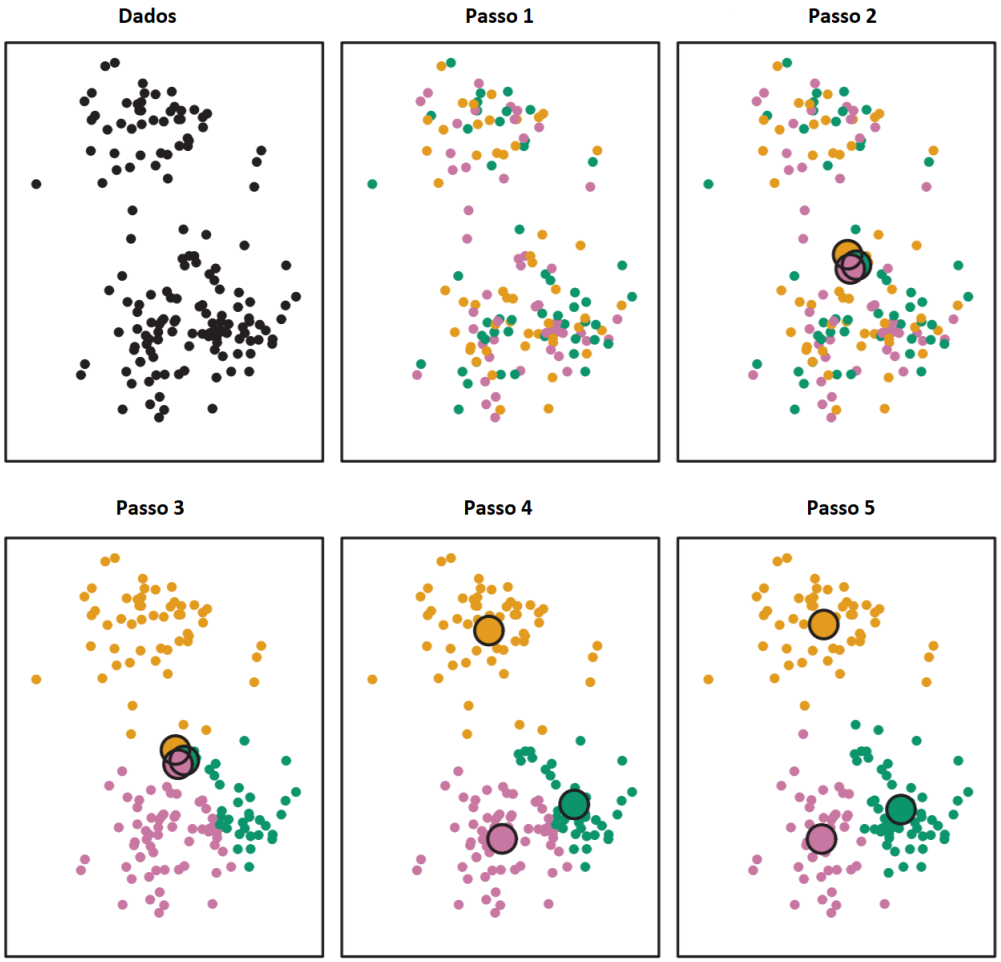


Figura 2: Progresso da execução do algoritmo de clusterização K-Means para K = 3.

Apesar do algoritmo K-Means possuir uma simples implementação e requisito computacional reduzido em comparação a outros algoritmos de clustering, uma desvantagem é o fato de que obtém uma solução ótima local, o que pode ou não implicar ser um clustering global ótimo. Nessa abordagem, os centroides aleatoriamente definidos inicialmente são utilizados para a obtenção da melhor solução possível para essa configuração inicial (solução ótima local), de forma que pode existir uma melhor solução se os centroides iniciais mudarem. Portanto, o sucesso do algoritmo K-Means está também muito vincu-

lado à definição aleatória dos centroides. Esse problema pode ser solucionado por meio da execução do algoritmo em diferentes pontos iniciais centroides, selecionando o melhor resultado obtido como o ponto ótimo global, limitando a vantagem de menor requisito computacional do algoritmo.

2.4 TESTE DA SOMA DAS ORDENS DE WILCOXON

Para a análise dos resultados obtidos e que são apresentados no capítulo 7, utiliza-se também o *Wilcoxon's Rank Sum Test* ou Teste da Soma das Ordens de Wilcoxon [21]. Trata-se de um teste de hipótese não-paramétrico indicado para comparação de dois grupos não pareados. Essa escolha baseou-se no fato de nem todos os dados a serem testados terem distribuição normal.

Para o cálculo do Teste da Soma das Ordens, consideram-se dois conjuntos de valores, denominados A e B, a serem comparados. Primeiramente, requer-se que os valores destes dois conjuntos sejam unidos e ordenados do menor para o maior valor. Em seguida, para cada valor é atribuído um número de posição, assim, ao menor valor atribui-se o número de posição 1, ao segundo menor valor atribui-se o número de posição 2 e assim sucessivamente. Caso haja valores repetidos, o número de suas posições são somados e o resultado da soma dividido pelo número de repetições existentes, sendo o número resultante deste cálculo atribuído a cada uma das repetições. Finalizada a atribuição do número das posições, realiza-se a somatória dos números da posição dos valores de cada um dos grupos, que correspondem respectivamente a W_A e W_B. A partir do menor valor entre W_A e W_B, calcula-se o valor de Z para determinar se a diferença entre os dois grupos é estatisticamente significante. Z é calculado pela expressão:

$$z = \frac{W - \mu_W}{\sigma_W} \tag{2.1}$$

μ_W e σ_W representam, respectivamente, a soma esperada e o desvio padrão de W, cujas expressões são:

$$\mu_W = \frac{n_A\,(n_A + n_B + 1)}{2} \tag{2.2}$$

Utiliza-se então o valor calculado de Z para obter o valor de p através da tabela de distribuição normal. Caso o valor obtido de p seja menor ou igual ao valor de significância

(α) estipulado, $H0$ é rejeitado, aceitando-se $H1$. Em caso contrário, $H0$ não é rejeitado [22, 23].

σ_W pode ser calculado por:

$$\sigma_W = \sqrt{\frac{n_A n_B \left(n_A + n_B + 1\right)}{12}} \tag{2.3}$$

3 TECNOLOGIAS UTILIZADAS

3.1 ROBOCUP RESCUE SIMULATION

O RoboCup Rescue Simulation é um projeto de pesquisa e educação iniciado como resposta a um desastre natural real, o terremoto que atingiu a cidade de Kobe no Japão em 17 de janeiro de 1995, matando mais de sessenta mil pessoas. A missão da Liga RoboCup é estimular a pesquisa na área, em que a implementação efetiva dessa missão é sumarizada em três principais pontos. O primeiro consiste em desenvolver um simulador capaz de representar desastres naturais de forma realista, visando prover planos de ação efetivos que resultem em menores impactos negativos na sociedade em diferentes cenários. Para isso, o simulador implementa diferentes entidades como ambulâncias, bombeiros e forças policiais para atuarem em uma cidade após um terremoto, com o objetivo de salvar o maior número possível de civis, sendo necessário para isso o desbloqueio de vias impedidas por escombros e o controle dos incêndios. O próximo ponto da implementação é definir parâmetros de desempenho que deem suporte a decisões reais em situações de desastres naturais. Por fim, busca-se promover a pesquisa e o desenvolvimento por meio da organização de competições que estimulem a troca de ideias e experiências entre pesquisadores.

O simulador do ambiente funciona como um servidor, ao qual o sistema desenvolvido se conecta enviando as ações a cada passo (*step*) uma vez que a conexão esteja estabelecida. Logo após esse estabelecimento, existe um tempo de pré-computação do ambiente, em que é permitido ao sistema analisar o mapa e traçar estratégias antes que a simulação inicie. Os agentes da simulação interagem com o ambiente e recebem *feedbacks*, sendo possível a comunicação entre si e que cooperem na realização de tarefas. Por exemplo, no caso de um resgate, duas ambulâncias em atuação conjunta conseguem finalizar a tarefa mais rapidamente do que no caso de atuação de apenas uma delas.

O RoboCup Rescue Simulation foi construído na plataforma Java (OpenJDK 8+) e roda no Sistema Operacional Linux (Ubuntu 16.04 LTS 64bit ou superior). O time de agentes também foi desenvolvido na mesma plataforma, seguindo um modelo que a

RoboCup nomeou Agent Development Framework (ADF), apresentado na seção 3.2. Essa estrutura, desenvolvida também em Java, engloba agentes de exemplo já com funções básicas de cada tipo.

O simulador calcula ao longo das execuções das estratégias dos times o score da partida, esse que é ponderado por fatores como número de civis sobreviventes, soterrados, feridos e dano aos edifícios. Quanto maior o score, melhor o desempenho do time. Adicionalmente, outros aspectos podem ser considerados na avaliação do desempenho dos times, a exemplo do tempo médio para resgate de um ferido e quantidade de civis identificados.

3.1.1 ENTIDADES

As entidades que compõem o simulador são: bloqueios, prédios (edifícios, Estação de Policiais ou Bombeiros, refúgio e Centro de Ambulâncias), ruas e agentes (civis, ambulâncias, bombeiros e policiais).

Na Figura 3 é apresentada a legenda das principais entidades do simulador.

Na Figura 4 é apresentado um dos mapas, o Kobe, além da disposição das entidades presentes em um instante da simulação.

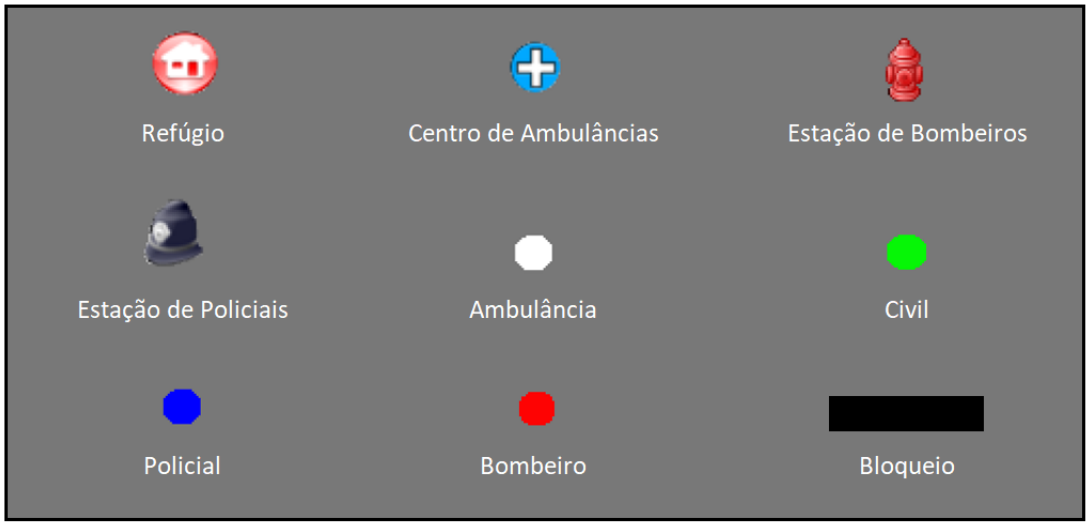


Figura 3: Símbolos das entidades da simulação.

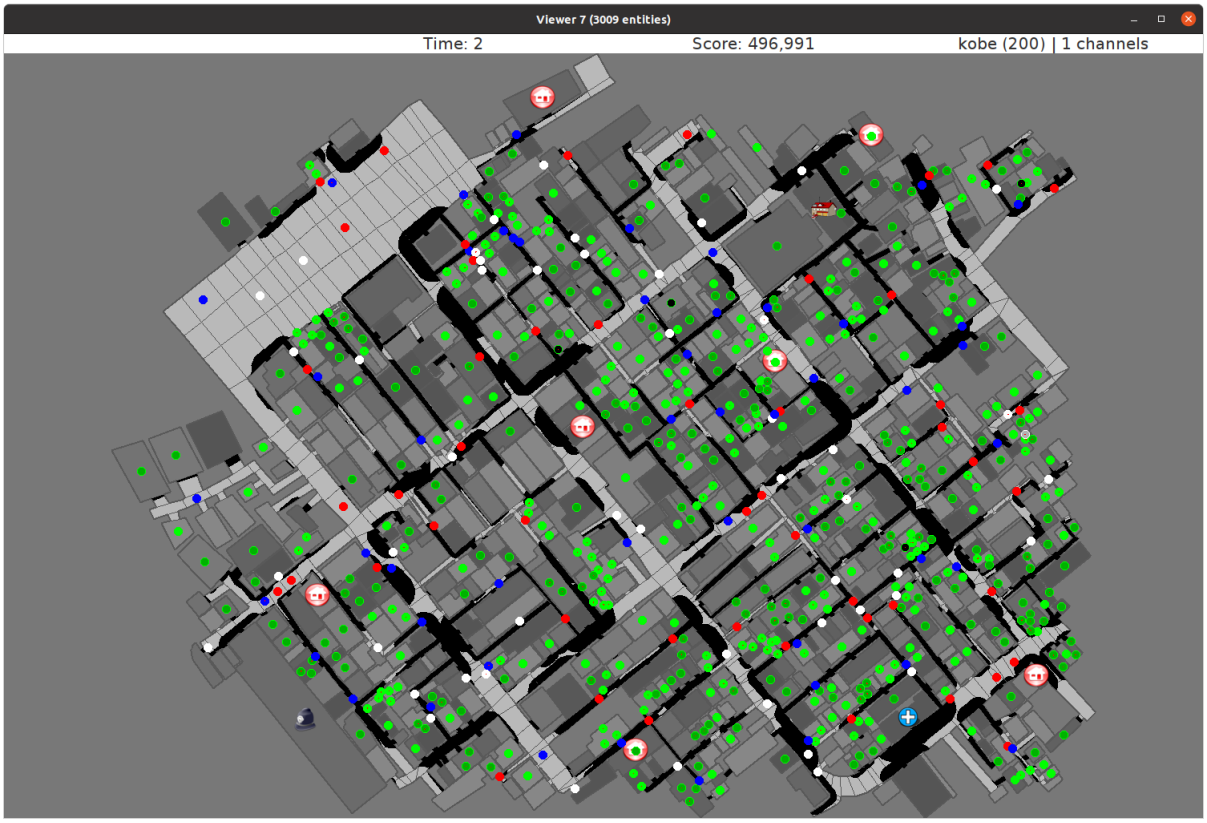


Figura 4: Exemplo de simulação do mapa Kobe do RoboCup Rescue Simulation.

3.1.1.1 OBSTÁCULOS

Os obstáculos são representados no simulador por polígonos pretos nas ruas, e obstruem a passagem de agentes e civis. São inseridos apenas no início da partida e caso exista o interesse em removê-los, o time policial deve ser acionado.

3.1.1.2 PRÉDIOS

Referem-se a todos os tipos de construções presentes no simulador, ou seja, edifícios, refúgios, quartéis de bombeiros e delegacias. Todos são representadas por polígonos de cores correspondentes aos seus estados: quanto mais escura a cor, maiores os danos estruturais decorrentes de incêndio ou vazamentos de água. Em especial, quartéis, refúgios e delegacias têm ícones próprios de identificação, e são imunes ao fogo.

No início da simulações, edifícios destruídos possuem vítimas presas nos escombros que devem ser removidos pelo time de resgate para salvamento dos feridos.

O refúgio representa um local destinado ao suporte médico ao resgatados e oferecimento de água aos Bombeiros. Na simulação, os resgatados que estiverem dentro do

refúgio não apresentam queda na sua pontuação de saúde. Enquanto os Bombeiros estiverem dentro do refúgio, periodicamente os reservatórios de água são repostos.

3.1.1.3 RUAS

As ruas são destinadas ao trânsito de pedestres e veículos, podendo estarem interrompidas por obstáculos.

3.1.1.4 AGENTES

Os agentes representam os civis, equipes de ambulância, bombeiros e policiais. Todos são representados por círculos de cores diferentes. Quanto menor a saúde, mais escura a cor. Os falecidos são representados pela cor preta.

Os civis não fazem parte do time de resgate, e portanto não são programáveis no simulador. São representados pela cor verde e o comportamento padrão deles é andar em direção ao refúgio mais próximo caso não estejam soterrados ou feridos. Caso se enquadrem em uma das duas situações referidas, devem ser transportados por uma ambulância.

O time de resgate é composto por ambulâncias, bombeiros e policiais. Podem ser divididos em dois grupos: agentes de pelotão e agentes centrais.

Os que fazem parte do time de ambulâncias são responsáveis por salvar agentes, levando civis aos refúgios ou removendo ambulâncias, policiais e bombeiros que estão soterrados. São capazes de remover vítimas de escombros e carregar uma pessoa no colo. Neste sentido, fazem parte do único tipo de time que se beneficia da colaboração de mais agentes, uma vez que cada vítima soterrada conta com um parâmetro de *buriedness*, nível de soterramento, que decresce a cada ação *rescue* realizada sobre a vítima. Esse decrescimento aumenta linearmente com a soma de esforços, diminuindo em valor com o mesmo número de ambulâncias resgatando a cada step.

O time de bombeiros é responsável por apagar incêndios em construções e carregam para isso água em seus tangues que podem ser recarregados em um refúgio.

Os policiais são responsáveis por remover obstáculos das ruas. Diferentemente dos bombeiros e ambulâncias, a atuação paralela de mais de um policial é ineficaz, não removendo um obstáculos mais rapidamente com a atuação de dois policiais em conjunto.

3.1.2 SEED

Os mapas possuem uma componente de aleatoriedade definida no arquivo *common.cfg* pela seed. Esse valor é responsável pela geração de obstáculos no mapa, de forma que alterações no valor da variável implicam em mudanças no posicionamento das barreiras. No Apêndice B pode-se verificar um exemplo para o mapa Kobe.

3.1.3 COMUNICAÇÃO NO SIMULADOR

Existem duas formas de comunicação no ambiente : comunicação direta e comunicação via rádio. A direta é realizada por meio do comando *speak*, e é audível em um determinado raio do emissor. Já a via rádio utiliza o comando *tell*, transmitindo a informação para todos os agentes que fazem parte do canal em que a mensagem foi direcionada. Os canais são limitados e cada um possui um limite de tráfego de dados. Existe um canal para cada tipo de agentes: bombeiros, ambulâncias e policiais.

Em ambos os tipos de comunicação a mensagem é transformada em uma sequência de bytes antes de ser enviada, e o receptor deve decodificar a mensagem assim que recebê-lá. Portanto, as mensagens são suscetíveis a erros, com mensagens podendo chegar em branco ou não sendo recebidas para representar as imperfeições no meios de comunicação nos ambientes de desastre.

3.2 AGENT DEVELOPMENT FRAMEWORK

O Agent Development Framework (ADF) é uma arquitetura de agentes altamente modularizada cujo uso é mandatório para todos os times da competição. Essa arquitetura é granularizada para cada processamento de dados ou atividades de decisões e divida em 5 módulos: alocação dinâmica de recursos, localização multi-agente de rotas, compartilhamento de informações, coleta de dados e formações de agrupamentos [24].

O ADF apresenta uma arquitetura de software bem definida que facilita a concentração dos esforços do participantes apenas no problema desejado, abstraindo complexidades que não são relevantes para a otimização dos algoritmos. Essa utilização padronizada do ADF favorece a comparação de resultados entre os times, possibilitando uma avaliação coerente de resultados obtidos. Adicionalmente, todo o código fonte do ADF é aberto, favorecendo reutilização dele para desenvolvimento de ideia próprias e comparação com resultados passados.

3.3 MAPAS

O ambiente em que os agentes realizam as ações foram criados com base informações de diferentes localidades do mundo utilizando o OpenStreetMap, o Geographic Markup Language (GML) e o Java OpenStreetMaps Editor (JOSM). O RoboCup Rescue Simulator inclui mapas de exemplo que podem ser utilizados para testes ou avaliação da performance dos algoritmos desenvolvidos.

3.3.1 CRIAÇÃO DOS MAPAS

3.3.1.1 OPENSTREETMAP

O OpenStreetMap (OSM) é uma plataforma gratuita de mapas disponível na web e que foi desenvolvida de forma colaborativa. A plataforma disponibiliza informações geográficas de diferentes partes do mundo gratuitamente e foi criada pelo projeto Planet OSM com o objetivo de prover informações geográficas em arquivos XML, apresentando os Nodes, Ways e Relations.

Os nodes representam pontos específicos no mapa e são definidos pela sua latitude e longitude. Cada node possui um identificar e um par de coordenada, podendo representar pontos no mapa como, por exemplo, uma cadeira em um parque.

Ways são listas ordenadas que possuem de 2 a 2.000 nós e que definem um polígono. São utilizados para presentar estradas, rios ou prédios, por exemplo.

Relations são estruturas de dados de uso versátil e que documentam a relação entre dois ou mais elementos (nodes, ways ou até mesmo outras relations).

3.3.1.2 GEOGRAPHIC MARKUP LANGUAGE

O Geographic Markup Language (GML) é uma linguagem baseada em XML utilizada para descrever informações geográficas e que foi definida pelo Open Geospatial Consortium. A ferramenta é amplamente utilizada para definições nos mapas, sendo utilizada no RoboCup Rescue Simulator para representar as entidades nos mapas.

3.3.1.3 JAVA OPENSTREETMAP EDITOR

O Java OpenStreetMap Editor (JOSM) é editor de mapas gerados pelo OpenStreetMap baseado em Java. O JOSM pode ser utilizado para download, edição e conversão de

mapas do formato OSM para o GML.

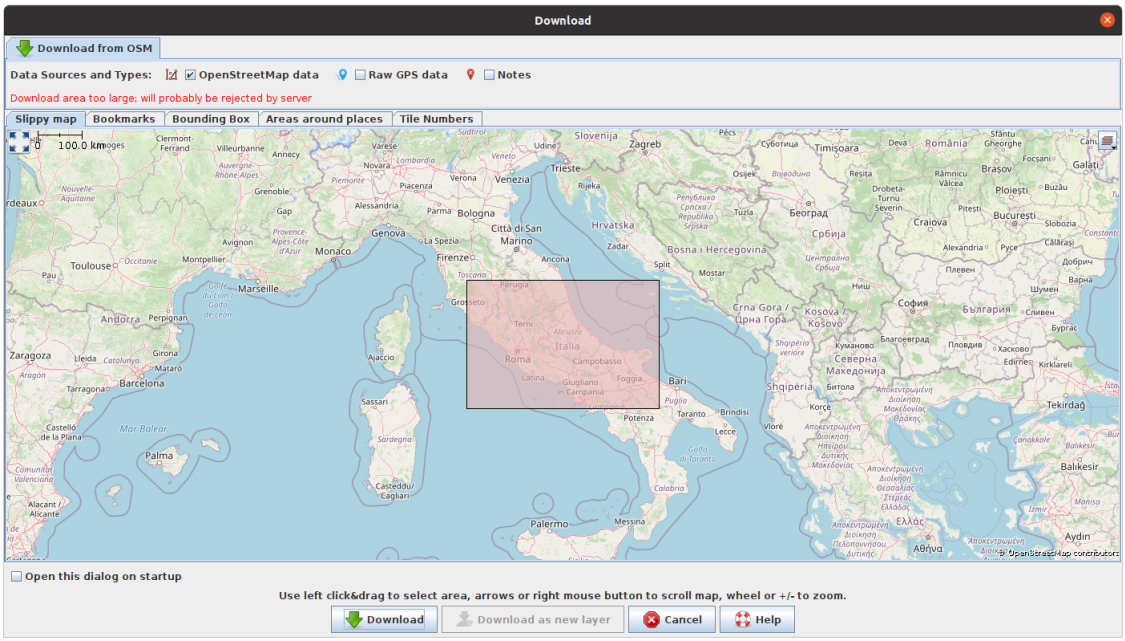


Figura 5: Interface do JavaOpenStreetMap editor com seleção de uma região da Itália no mapa.

3.4 VISUAL DEBUGGER

O Visual Debugger [25] foi desenvolvido pelo time iraniano MRL no ano de 2016 com o objetivo de adição de camadas de visualização para facilitar o monitoramento do comportamento das entidades da simulação e suas respectivas execuções de tarefas e alvos. Com isso, a avaliação de resultados de diferentes algoritmos torna-se mais fácil, assim como o processo de correção de eventuais erros no código.

Considerando a natureza do RoboCup Simulation, o monitoramento da performance e comportamento dos agentes pode ser mais fácil pelas respostas visuais do programa ao invés de logs baseados em textos. Diante desse cenário, o time MRL desenvolveu a ferramenta que tem uma interface similar ao visualizador oficial da competição, adicionando funcionalidades gráficas para prover representações em tempo real dos dados.

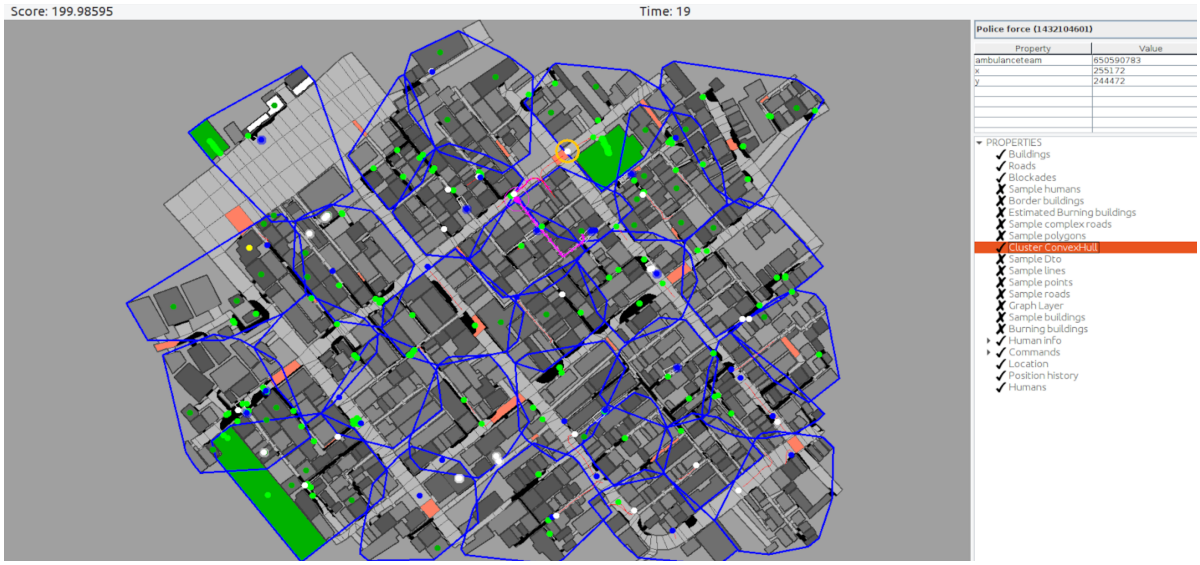


Figura 6: Tela principal do Visual Debugger.

O Visual Debugger também é baseado em Java e atua com duas categorias de visualização na simulação: a representação de dados no mapa como um todo (como a ilustração dos clusters do mapa) ou a adição de dados para cada uma das entidades (como novas informações dos civis) [26].

3.5 BASH SCRIPT

O Bash (Bourne Again Shell) é um interpretador de comandos que foi escrito originalmente para o projeto GNU. O interpretador Bash define o Bash Script, a linguagem de programação que foi utilizada no desenvolvimento deste trabalho com foco na coleta de dados das simulações [27].

Algumas operações típicas executadas por linguagens de script em Shell incluem; manipulação de arquivos, execução de programas e impressão de texto. Sendo assim é muito comum ver scripts sendo utilizados para automatização de tarefas, dado que os arquivos de script permitem construir esquemas de execução complexos a partir dos comandos básicos do shell.

No Capítulo 6 é apresentada a implementação do script para tomada automatizada dos resultados obtidos pelo algoritmo implementado neste trabalho.

3.6 BIBLIOTECA SCIPY

O SciPy é uma coleção de algoritmos matemáticos e funções que estender as capacidades da biblioteca NumPy no linguagem de programação Python. O SciPy adiciona funções e algoritmos para uma grande variedade de tarefas, incluindo quantização de vetores, funções estatísticas, transformadas de Fourier, regressões de distância ortogonal e ferramentas de interpolação [28], por exemplo.

O foco da utilização dessa biblioteca neste trabalho é a função *wilcoxon*, responsável por implementar um teste de Wilcoxon por meio do teste da hipótese nula na qual dois pares de dados derivam da mesma distribuição probabilística. Particularmente, a função teste se a distribuição das diferenças entre as duas amostrar é simétrica e próxima a zero, sendo uma versão não paramétrica do teste T, assim como apresentado em 2.4.

A função é definida por:

```
scipy.stats.wilcoxon(x, y=None, zero_method='wilcox', correction=False,
↪  alternative='two-sided', mode='auto')
```

A variável x representa o primeiro conjunto de dados e y o segundo, sendo ambos unidimensionais. Como buscando implementar o Teste Mann-Whitney-Wilcoxon, a variável zero method deve ser mantida como 'wilcox'.

4 METODOLOGIA DO TRABALHO

A segmentação lógica da Metodologia do Trabalho é composta por cinco etapas: revisão de literatura disponível, definição da concepção do projeto, definição dos critérios de avaliação, depuração e por fim avaliação de resultados obtidos.

4.1 REVISÃO DE LITERATURA

A literatura relacionada é bastante ampla, com a primeira competição e publicação ocorrendo em 2001. Um dos aspectos pesquisados foi a alocação de tarefas aos agentes, já que os mesmos possuem conhecimento apenas parcial de incidentes ocorrendo no momento, sejam eles focos de incêndio, ruas bloqueadas ou feridos. Dessa forma, uma decisão a ser feita é se devem buscar novas tarefas ou não, e qual a priorização de cada uma dessas tarefas em um determinado momento.

Outro aspecto já abordado foi a formação de grupos de salvamento de civis, uma vez que a combinação de esforços médicos é frequentemente necessária no salvamento dos feridos, seja por meio da cooperação entre forças policiais que necessitam liberar ruas bloqueadas que ligam o local do acidente aos hospital e médicos que dão o suporte às vítimas ou a necessidade de uma atuação conjunta do Corpo de Bombeiros para apagar focos de incêndio e médicos socorristas.

Adicionalmente, um outro exemplo de tópico já abordado na literatura é a comunicação entre agentes, em que se pode utilizar o canal de rádio para permitir tomadas de decisões em conjunto com base na troca de informações, além da possibilidade de criação de novos modelos de comunicação entre agentes.

Com o suporte do Orientador e Co-orientador deste trabalho, este último sendo membro do Comitê Executivo do RoboCup Rescue, verificou-se que ainda não foram realizadas abordagens que buscam atuar na setorização dinâmica e particionamento lógico da área de resgate em que se encontram os civis e a equipe de salvamento. As que chegam a utili-

zar setorização dividem o mapa sem considerar informações adicionais como a densidade de prédios em cada região. Essa informação pode ser de grande valor, dado que regiões com maior quantidade de prédios requerem também uma quantidade superior de agentes para explorarem a área.

4.2 CONCEPÇÃO DO PROJETO

Pelo fato das abordagens realizadas nas competições e levantadas na literatura segmentarem as áreas estaticamente, optou-se por estudar os efeitos que a segmentação dinâmica do mapa, já que novas informações adquiridas pelos agentes são essenciais para se fazer novas setorizações.

O desenvolvimento inicia-se com foco em dois agentes de resgate: ambulâncias e policiais. A complexidade do projeto requer a evolução gradual dos desafios, e uma abordagem paralela dos três agentes de resgate dificultaria a obtenção de resultados claros do andamento do projeto.

Pelo fato da equipe médica não conseguir atuar sem o auxílio policial, uma vez que não conseguiriam se deslocar até os feridos sem auxílio para liberação das ruas bloqueadas, optou-se por dar prioridade à essas duas entidades dos agentes de resgate. Portanto, os bombeiros são foco posterior, agregando complexidade ao projeto de forma conveniente.

Além da programação de apenas duas entidades no início do projeto, para a segmentação do mapa ocorrer de forma sistêmica e eficiente, o estudo dos parâmetros que caracterizam os mapas da simulação torna-se importante. Como exemplos de parâmetros, tem-se a característica do relevo da região, altura média dos edifícios, idade da população local, proximidade dos focos de vítimas dos refúgios, quantidade de focos de incêndio e quaisquer outros parâmetros que possam caracterizar o ambiente.

Para a setorização ocorrer, buscam-se parâmetros que possam descrever áreas que possuam determinadas similaridades. Como exemplo, um edifício que está próximo a desabar e que possui um elevado número de civis pode ser considerado um setor, assim como uma região pouco densa no quesito populacional pode ser considerada outro setor.

Cria-se assim uma matriz que atribui graus de risco a cada uma das áreas e pode-se setorizar essas regiões como forma de definir as melhores estratégias para socorro das vítimas.

4.3 CRITÉRIOS DE AVALIAÇÃO

Para comparação de resultados e avaliação estatística, são avaliados dois principais grupos de métricas: o score das simulações e informações atreladas aos resgates dos civis. O algoritmo implementado busca maximizar ambos critérios.

O processo de medição de resultados foi dividido entre a coleta de dados e a avaliação estatística, com apresentação dessas informações no capítulo 7. Nas subseções seguintes são definidas cada uma das métricas avaliadas.

4.3.1 SCORE DA SIMULAÇÃO

O simulador RoboCup Rescue define uma metodologia de cálculo dos scores das simulações baseada em duas principais métricas: saúde média dos civis e nível de dano físico nos prédios. O cálculo é definido por:

$$Score = HPscore * buildingScore \tag{4.1}$$

sendo

$$HPscore = agentsAlive + (hpLeft/hpMax) \tag{4.2}$$

e

$$buildingScore = \sqrt{areaLeft/areaMax} \tag{4.3}$$

Como neste trabalho o foco é a atuação das ambulâncias e os incêndios foram desabilitados, o score da simulação é essencialmente composto pelos resultados atrelados aos civis, dependente da quantidade de civis vivos no final da simulação e da vida restante dos civis.

4.3.2 MÉTRICAS DE RESGATE

Além do score, as seguintes métricas de resgate são avaliadas: civis identificados, civis designados, tentativas de resgate, civis carregados pelas ambulâncias, civis resgatados, steps até a identificação, steps de alocação, steps das tentativas, steps de carregamento dos civis e steps dos resgates.

4.3.2.1 CIVIS IDENTIFICADOS

Métrica que representa o total de civis identificados ao longo da simulação. A identificação não necessariamente implica em um resgate efetivo, uma vez que uma ambulância pode não ser capaz de realizar um resgate no caso do soterramento do civi ser grave, por exemplo.

4.3.2.2 CIVIS DESIGNADOS

Métrica que representa o total de civis que tiveram uma ambulância designada para o resgate. Uma ambulância pode ser incapaz de se aproximar do civil por conta dos obstáculos que precisam ser removidos pelas forças policiais.

4.3.2.3 TENTATIVAS DE RESGATE

Métrica que representa o total tentativas realizadas pelas ambulâncias no salvamento dos civis. Uma tentativa pode falhar também no caso da existência de obstáculos no caminho.

4.3.2.4 CIVIS CARREGADOS

Métrica que representa o total de civis que foram efetivamente carregados pelas ambulâncias. No caso de um civil soterrado, o carregamento só ocorre após a liberação do civil dos escombros.

4.3.2.5 CIVIS RESGATADOS

Métrica que representa o total de civis que foram deixados vivos nos refúgios pelas ambulâncias.

4.3.2.6 CIVIS MORTOS

Métrica que representa o total de civis que não sobreviveram até que chegasse o resgate das ambulâncias.

4.3.2.7 STEPS ATÉ A IDENTIFICAÇÃO

Número de passos da simulação necessários até a primeira identificação de um civil por uma ambulância.

4.3.2.8 STEPS DE ALOCAÇÃO

Número de passos da simulação necessários até a primeira alocação de uma ambulância ao resgate de um civil.

4.3.2.9 STEPS DAS TENTATIVAS

Número de passos da simulação necessários nas tentativas de resgates de civis.

4.3.2.10 STEPS DE CARREGAMENTO

Número de passos da simulação necessários até o primeiro carregamento de um civil em uma ambulância.

4.3.2.11 STEPS DOS RESGATES

Número de passos da simulação necessários até o primeiro resgate efetivo de um civil, deixando-o em um refúgio.

4.3.2.12 STEPS DOS MORTOS

Número de passos da simulação até um civil ser considerado morto.

4.4 DEPURAÇÃO

O processo de depuração foi realizado continuamente durante toda a implementação do projeto. Sugestões de melhorias na abordagem algorítmica foram apontadas nas reuniões periódicas em conjunto com o orientador Jaime Simão Sichmann e o co-orientador Luis Gustavo Nardin, membro do Comitê Executivo do RoboCup Rescue Simulation. As reuniões no início do projeto possuíam frequência quinzenal e foram intensificadas para semanais com o andamento do projeto.

Para o gerenciamento e versionamento de códigos utilizados e desenvolvidos pela equipe foi empregado o Git, hospedado no servidor de repositórios Azure Repos da Microsoft. Na mesma plataforma, criou-se um projeto no Azure DevOps, contendo as tarefas de código a serem implementadas e a organização do trabalho em questão de prazos e entregáveis.

A principal forma de depuração do código foi feita através de leitura de logs e visualização no Visual Debugger (seção 3.4), de forma conjunta. Devido à elevada quantidade de agentes atuando no ambiente, é praticamente impossível compreender o que acontece utilizando-se somente de um ou de outro de forma isolada. No contexto deste trabalho, o foco principal foi no log da Central de Ambulâncias, que faz a alocação de ambulâncias nos clusters e envia comandos a cada uma delas de acordo com informações recebidas em cada passo da simulação.

Em relação aos logs da Central de Ambulâncias, a cada passo imprime-se as informações que tem relativas às ambulâncias espalhadas pelo mapa. Exemplos dessas informações são: alvo designado, cumprimento ou não da ordem comunicada, avisos de término de ação, ações sendo executadas, civis que morreram, civis que foram resgatados com sucesso.

Já na parte visual, foram criadas novas camadas de informação, tornando mais rápida a identificação de alguns pontos centrais da simulação. A primeira camada desenvolvida diz respeito ao cluster ao qual o agente atualmente selecionado pertence. Nela, todas as construções pertencentes a este cluster são coloridas com a cor ciano. A segunda camada de informações mostra um círculo magenta ao redor de civis soterrados que foram identificados pela central. Outra alteração feita foi um círculo laranja ao redor de ambulâncias quando estas estão carregadas com uma vítima. Finalmente, na versão do time que contém diretivas de exploração, as construções que devem ser exploradas pela ambulância selecionada ficam pintadas em laranja.

4.5 AVALIAÇÃO DE RESULTADOS

Considerando que cada simulação pode levar até 30 minutos para ser executada, uma vez considerada a ampla quantidade de entidades envolvidas, desenvolveu-se um script em Bash para automatizar o processo de tomada de dados (seção 6.2). A simulação possui um componente de aleatoriedade definido por uma seed, de forma que alterações nesse valor implicam mudanças nas ruas bloqueadas da simulação. Considerando esse aspecto, foram

obtidos dados para o time de exemplo fornecido pelo RoboCup Rescue, assim como para os times desenvolvidos pelo grupo, com o algoritmo de clusterização K-Means e diferentes valores de K.

O script em Bash foi responsável por salvar tanto os dados gerados pelas simulações quanto o log da central, contendo informações como a pontuação, tempo para primeiro resgate de civil e quantidade de sobreviventes em um arquivo texto que foi posteriormente processado via Pandas do Python.

Os valores de K escolhidos foram (1, 2, 4, 8), buscando avaliar os impactos de diferentes quantidades de clusters na simulação. Tmabém foram escolhidos 5 mapas (Berlin, Joao, Kobe, Paris e Berlin2), melhores descritos na Seção 7.1. Além disso, para fidelidade estatística, foram feitas simulações com 10 valores diferentes de seed para cada K, time e mapa. Dessa forma, totalizam-se 450 simulações, sendo 400 dos dois times desenvolvidos pelo grupo, mais 50 simulações do time exemplo que será utilizado como baseline.

Para análisar os dados obtidos escolhemos o teste estatístico de Wilcoxon-Mann-Whitney. Também conhecido como teste da soma dos postos de Wilcoxon, este é uma alternativa não paramétrica para o Teste t para 2 amostras. O método escolhido, ao contrário do Teste t, não assume que as amostras são aleatórias, ou seja, seguem uma distribuição normal, ao contrário, ele não assume distribuição nenhuma. Para calcular estatisticamente Wilcoxon é atribuído um valor a cada um dos valores das duas amostras para construir ranking, e esses rankings têm suas distribuições comparadas. Sob a hipótese nula, a distribuição a partir de ambos os grupos é a mesma. Sob a hipótese alternativa, os valores das amostras tendem a exceder os dos outros. O p-value escolhido para constatar diferença significante estatisticamente foi 0.05, valor padrão para tal análise. Mais profundamente explicado na Seção 2.4

5 ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

Nesse capítulo são abordados tanto os requisitos funcionais como não funcionais do algoritmo do time desenvolvido. Adicionalmente, a competição RoboCup Rescue Simulation define a metodologia que deve ser seguida no desenvolvimento dos algoritmos dos times, e decisões de implementação consideraram esses fatores. Por fim, aspectos técnicos da implementação do ADF foram considerados na definição de requisitos dos sistema.

5.1 DESCRIÇÃO DO SISTEMA

O objetivo principal do funcionamento do sistema é criar um time que utiliza o algoritmo K-Means de clustering implementado na central de ambulâncias para fazer uma alocação de agentes que considera a área dos prédios de cada cluster. Dessa forma, busca-se maximizar os sobreviventes do acidente assim como reduzir o tempo de salvamento de cada civil, e consequentemente melhorar o score da simulação. Esse particionamento deve ser realizado pela central de ambulâncias no início da simulação, designando os clusters para cada uma das ambulâncias da simulação.

Pelo fato do time desenvolvido estar inserido na simulação, o sistema deverá ser capaz de lidar com os desafios impostos por uma crise, que são simulados pelo servidor. Desafios incluem desabamento de prédios por danos decorrentes do terremoto e também os bloqueios nas vias, requerendo um trabalho conjunto entre equipes policiais e ambulâncias, por exemplo. No início da simulação, o mapa deverá ser explorado com base na clusterização realizada na Central de Ambulâncias, e as informações adquiridas por cada agente deverá ser compartilhada entre eles, inicialmente passando as informações para a central correspondente, e então entre centrais. Ao longo da simulação, planos de ação devem ser criados e seguidos, levando em conta as informações adquiridas inicialmente e também as que forem sendo descobertas por cada agente.

5.2 REGRAS PARA O DESENVOLVIMENTO DO TIME

Para a criação do time de agentes usando o Agent Development Framework (ADF), deve-se utilizar a estrutura do repositório *rcrs-adf-sample* como base. Os arquivos do diretório *src/adf* não podem ser alterados no desenvolvimento. O diretório do novo time deve ser criado na pasta *src*, assim com o do diretório de exemplo fornecido e chamado *test_team*.

Dentro do diretório do time criado, devem ser implementadas as classes que substituirão as classes do agente fornecido *sample*.

Não é permitido alterar qualquer classe do agente *sample*, é permitido alterar apenas as classes que estão nos seguintes diretórios:

```
src/adf/sample/centralized
src/adf/sample/extraction
src/adf/sample/module
src/adf/sample/module/algorithm
src/adf/sample/module/complex
```

Nesses arquivos desses diretórios as alterações são livres, inclusive sendo liberado alterações no nome da classe. O único aspecto que deve ser mantido é a assinatura da classe, já que a mesma define os métodos que permite a interação do framework com a nova classe.

5.3 REQUISITOS ADICIONAIS DO SISTEMA

O sistema é composto pelo algoritmo K-Means para clustering e implementado da central de ambulâncias, realizando a distribuição dos agentes entre os clusterings já no início da simulação. O sistema deve ser capaz de utilizar outros algoritmos de clustering apenas por meio da alteração das chamadas de função dos arquivos da pasta Algorithmic do simulador.

A distribuição da quantidade de ambulâncias para cada cluster deve considerar a área dos prédios de cada região em relação à área total do mapa: quanto maior a área dos prédios dentro de um cluster, maior a probabilidade de civis feridos, requerendo portanto mais agentes para o resgate.

O sistema deve apresentar no simulador a visualização dos clusters definidos, possibilitando avaliar também de forma visual os resultados do algoritmo K-Means na clusterização.

6 PROJETO E IMPLEMENTAÇÃO

Este capítulo descreve as etapas de desenvolvimento do projeto e as decisões tomadas durante seu percurso. São discutidos alguns detalhes de implementação, de forma que o leitor possa compreender o funcionamento do algoritmo desenvolvido.

6.1 VISUALIZAÇÃO DOS CLUSTERS

Para possibilitar a visualização dos clusters criados o grupo teve apoio de Madhi, desenvolvedor iraniano do time MRL que participa das competições do Robocup Rescue Simulator. O código foi estudado e alterado para enviar as informação ao debug viewer do simulador, permitindo visualização gráfica da clusterização.

O visualizador destaca na cor azul os clusters, cuja quantidade é definida no código do K-Means. Pode-se perceber que os clusters ficam centralizados em áreas densas de prédios, onde a probabilidade de civis feridos é maior.

Para avaliação estatística dos resultado da clusterização, é necessário coletar resultados seguidamente com diferentes quantidades de clusters e em diferentes mapas do simulador, dado que cada mapa possui uma particularidade: menos prédios, mapas mais extensos, menos densos ou com menos civis. Dessa forma, para automatizar a coleta de dados criou-se um script em BASH que permite a coleta de dados automática, bastando definir os parâmetros que devem avaliar. Dentre os parâmetros variados, destaca-se o valor de K (quantidade de clusters), os seeds de aleatoriedade e os mapas.

Para o algoritmo de K-Means, a implementação na central de ambulâncias foi crucial para permitir que a alocação fosse realizada considerando a área total dos prédios do mapa, atribuindo uma quantidade maior de agentes nas áreas com maior densidade de prédios.

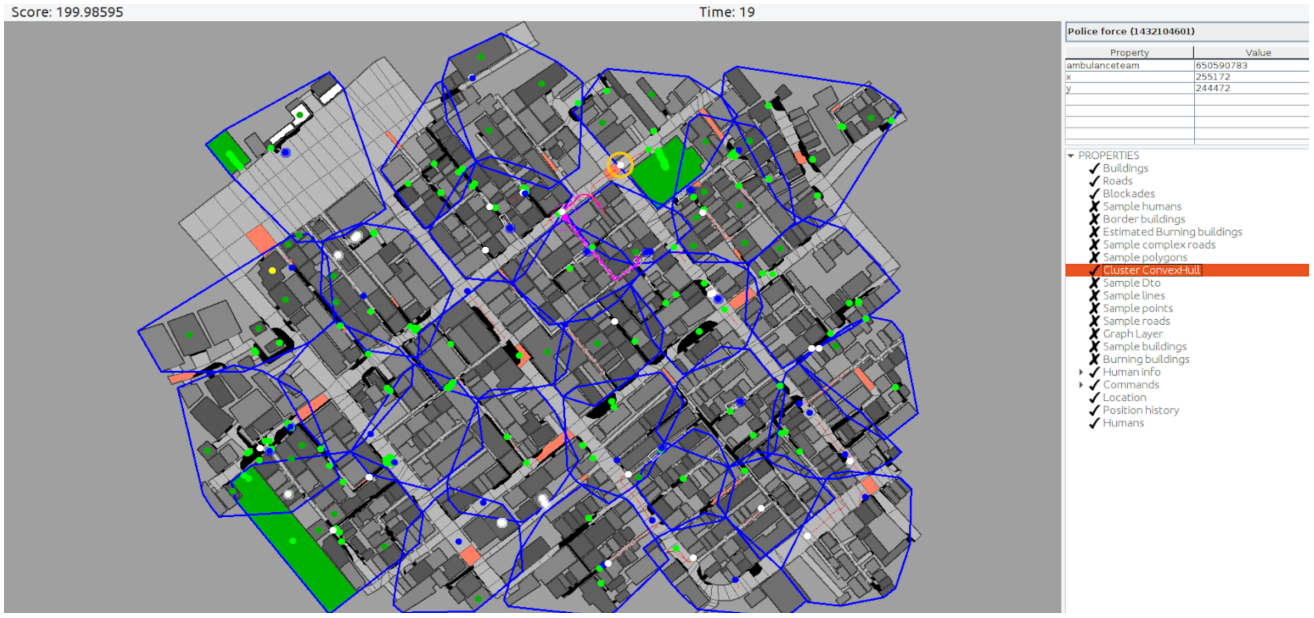


Figura 7: Visualizador de clusters

6.2 BASH SCRIPT

Para que fosse possível fazer todas as simulações necessárias para se tirar conclusões a respeito das hipóteses levantadas, a saber a melhora de pontuação decorrente da utilização de clusters e da centralização de comandos, foi desenvolvido um script em bash. Maiores detalhes de como foi estruturada a decisão do número de simulações necessárias estão dispostos na seção 4.5. O script é parametrizável, utilizando-se de um arquivo de configuração contendo os parâmetros de execução. Estes parâmetros são mapas, número de clusters K e seeds. Além disso, neste arquivo de configuração também devem constar os caminhos referentes a onde salvar os logs da simulação, onde estão localizados os mapas e onde estão sendo gerados os logs do servidor de simulação. Finalmente, contém também uma flag para debug, que, se tiver definida como verdadeira, indica para o servidor que o Visual Debugger dever ser iniciado com cada simulação.

A entrega de resultados do script segue uma ordem, organizada a fim de facilitar a compreensão da arvore de diretórios produzida. Desta forma, em primeiro lugar, cria um diretório para o mapa sendo simulado, e então, para cada K fornecido como parâmetro, cria uma subpasta cujo nome é o próprio valor de K. Dentro desta subpasta, cria então mais uma pasta para cada seed diferente de simulação.

A sua execução foi organizada para reduzir o número de recompilações necessárias do agente, e pode ser descrita no seguinte pseudo-código:

```
for K in ${CLUSTERS[*]};
do
  # Mudar a classe de constantes do time, trocando o valor de K no código
  # Limpar o projeto e recompilá-lo
  for MAP in ${MAPS[*]};
  do
    for SEED in ${SEEDS[*]};
    do
      # Mudar a linha com a seed da simulação no servidor
      # Criar pasta no formato nome-do-time/mapa/k/seed
      # Se já existe, pular a execução
      # Iniciar o servidor
      # Se a flag debug estiver ativa, iniciar Visual Debugger
      # Iniciar time
      # Esperar o fim do simulador
      # Fechar o agente
      # Copiar saída do simulador para a pasta de logs passada nos parâmetros
      # Fechar Visual Debugger se estiver aberto
      # Extrair linhas que falam sobre a pontuação do arquivo de log principal
    done ;
  done ;
done ;
```

A última linha de extração de pontuação foi incluída devido ao grande tamanho do arquivo gerado pela simulação, podendo chegar a 5 GB por simulação em alguns casos. A implementação pode ser consultada no Apêndice A.

6.3 K-MEANS

Abaixo a sequência de tarefas a serem executadas pela Central de Ambulâncias e pelas Ambulâncias:

1. A Central de Ambulâncias executa o particionamento do mapa usando o algoritmo K-Means. O número de partições pode depender do número de Ambulâncias no cenário ou fixado por meio de configuração;
2. A Central de Ambulâncias identifica a localização das Ambulâncias;

3. A Central de Ambulâncias aloca as Ambulâncias para uma partição específica, por exemplo, a partição com centróide mais próxima a cada Ambulância;
4. A Central de Ambulâncias informa as Ambulâncias as partições e a partição designada para cada uma delas;
5. As Ambulâncias exploram aleatoriamente os prédios na partição designadas a elas;
6. Quando a Ambulância identifica alguma vítima, ela envia essa informação para a Central de Ambulâncias;
7. A Central de Ambulâncias consolida a informação das vítimas enviadas pelas Ambulâncias;
8. A Central de Ambulâncias define qual a tarefa que cada Ambulância deve realizar;
9. A Central de Ambulâncias envia a tarefa a ser desempenhada pelas Ambulâncias para as Ambulâncias;
10. A Ambulância finaliza a tarefa a ser realizada e informa a Central de Ambulâncias que está livre.

Os itens 1-4 são realizados somente na inicialização da Central de Ambulâncias. Os itens 5-9 são realizados todos os ciclos de simulação. O item 10 é realizado quando a Ambulância termina a tarefa designada

Se uma Ambulância recebeu uma tarefa, a Central de Ambulâncias não pode definir uma nova tarefa para aquela Ambulância até que a Ambulância envie a confirmação de que ela está livre.

Devido à extensão dos arquivos modificados na implementação dessas tarefas, os arquivos foram disponibilizados no Azure DevOps e estão disponíveis no seguinte link: https://dev.azure.com/Poli-Robocup/_git/Robocup%202020.

6.4 TIMES

Ao se desenvolver a primeira implementação de time com a alocação de ambulâncias por clusters, percebeu-se que era comum na simulação que ambulâncias saíssem do cluster designado para performar tarefas autonomamente. Isso causava um comportamento não esperado, em que podia-se ver exemplos de agentes cruzando o mapa para ajudar com

alguma tarefa e então, retornando ao seu cluster porque uma tarefa da central foi recebida. Esse tempo era "gasto"inutilmente, e decidimos, então, tentar uma abordagem diferente e comparar resultados.

Dessa forma, uma nova abordagem foi tomada na implementação, em que, além de serem alocadas para alvos que só estivessem no seu cluster, as ambulâncias também seriam ordenadas para fazerem uma exploração das construções do cluster a que pertencem. Uma vez que todas as suas construções tiverem sido varridas pelo menos uma vez, então os agentes estariam livres para serem alocadas a um civil que já estivesse sendo resgatado, somando esforços. Em outras palavras, a ordem de prioridade das ações se traduz em salvar civis que não estão sendo atualmente resgatados, explorar o cluster enquanto não for explorado por completo, ajudar em outros resgates, explorar o cluster novamente. Com essa abordagem, as ambulâncias não saem do cluster alocado em nenhuma hipótese, salvo em alguns intervalos entre término de ação e recebimento de novo comando.

À essa implementação foi dada o nome Time 2, e estará presente na análise comparativa de resultados. Com o objetivo de dar suporte à mudança, uma nova camada no Visual Debugger foi criada, com o nome *Scout Buildings*. As construções alocadas para serem exploradas por cada ambulância selecionada são coloridas em laranja.

7 TESTES E AVALIAÇÃO

Para avaliação dos resultados obtidos foram escolhidos inicialmente quatro mapas: Kobe, Paris, João Pessoa e Berlin. Para automatização foi criado um script em BASH para execução da simulação considerando os três times do estudo (Baseline, Time 1 e Time 2), quatro valores de K para os Clusters do KMeans e dez valores de seeds diferentes, componente responsável pelo o aspecto aleatório na geração dos mapas: o seeds impactam nas ruas do mapa que estão bloqueadas pelos desabamentos, estes bloqueios requerem uma autoridade policial para desimpedimento da via.

Em termos de *hardware*, na competição são utilizados 4 computadores pra cada time, um rodando o simulador e os outros três rodando o código dos agentes. Todos os computadores tem a especificação mínima de ser um processador Intel de pelo menos 4 núcleos com 8 GB de memória RAM.

Já na nossa simulação, foram utilizados dois computadores, ambos com no mínimo 4 núcleos e 8GB de memória RAM, totalizando 10 simulações distintas pra cada combinação de 3 Times, 4 Clusters e 5 Mapa, totalizando 450 simulações (pois o time Baseline só utiliza o KClusters = 1)

7.1 MAPAS ESCOLHIDOS

Com base nas tecnologias apresentadas, a liga RoboCup Rescue definiu mapas a serem utilizados na avalição dos resultados na competição, e como diretriz adotada desde a última competição oficial da RoboCup Rescue Simulation em 2019, os focos de incêndio foram desabilitados, visando tornar mais claro o efeito das decisões das forças policiais e ambulâncias. Neste trabalho foram utilizados 4 mapas oficiais da competição, buscando avaliar impactos de diferenças geográficas e de estratégia do algoritmo para mapas com características diferentes. Nas subseções seguintes cada um dos mapas é apresentado, assim como suas peculiaridades geográficas são apontadas.

7.1.1 KOBE

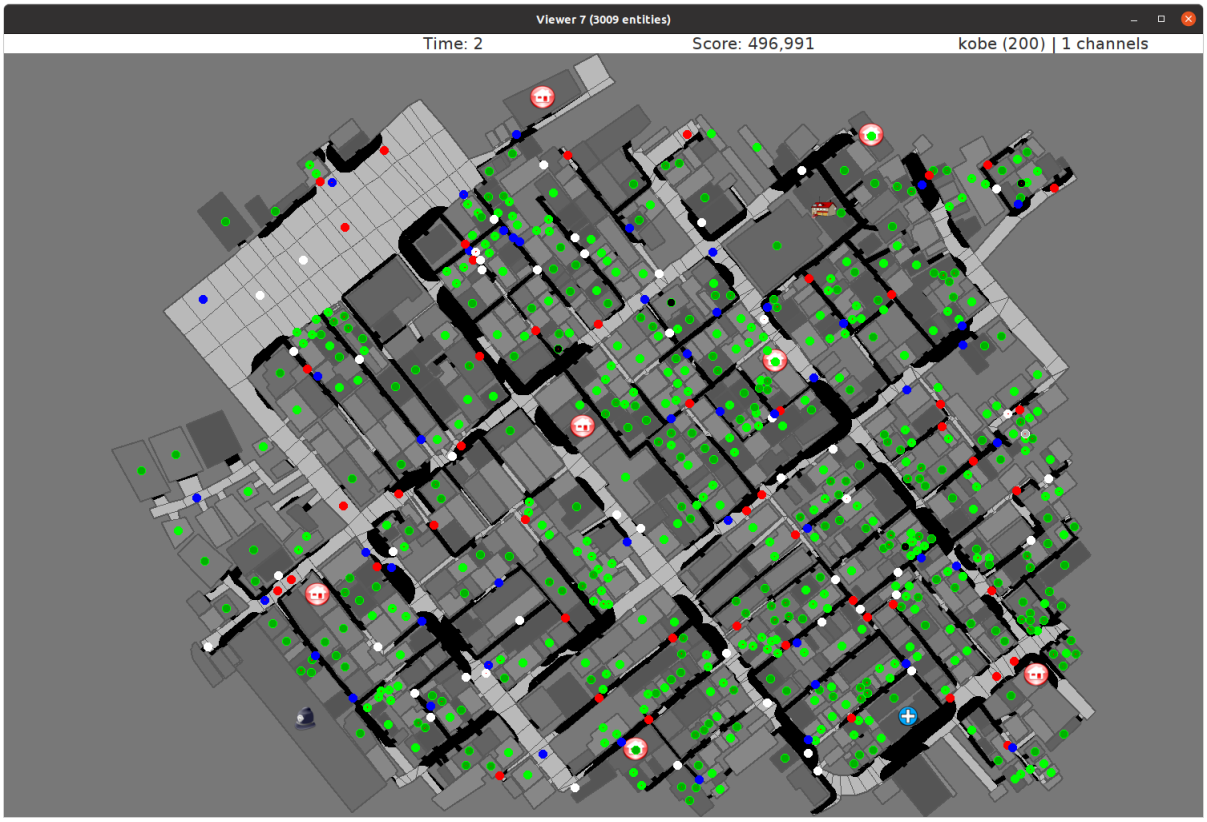


Figura 8: Mapa Kobe da RoboCup Rescue Simulation.

O projeto RoboCup Rescue Simulation iniciou como resposta ao terremoto que atingiu a cidade de Kobe no Japão em 17 de janeiro de 1995, matando mais de sessenta mil pessoas. Dessa forma, esse mapa é utilizado como mapa padrão da simulação.

Em relação aos outros 3 mapas que serão descritos a seguir, o Kobe possui distribuição relativamente uniforme de civis, assim como de áreas bloqueadas. O mapa possui 7 refúgios para civis também com distribuição uniforme.

7.1.2 BERLIN E BERLIN2

Figura 9: Mapa Berlin da RoboCup Rescue Simulation.

Berlin é o maior mapa da simulação, apresentando grande quantidade e densidade de civis, além de regiões afastadas nos cantos do mapa com concentrações de civis feridos. O mapa possui apenas 4 refúgios de civis e as muitas barreiras que precisam ser liberadas pelas forças policiais não são uniformemente distribuídas pelo mapa.

Berlin2 é a versão alterada deste mapa em que bloqueios não são gerados, tornando possível, assim, a locomoção livre de todos os agentes desde o primeiro step de simulação, mas mantendo os desabamento de prédios, sendo necessário resgatar o mesmo número de civis.

7.1.3 PARIS

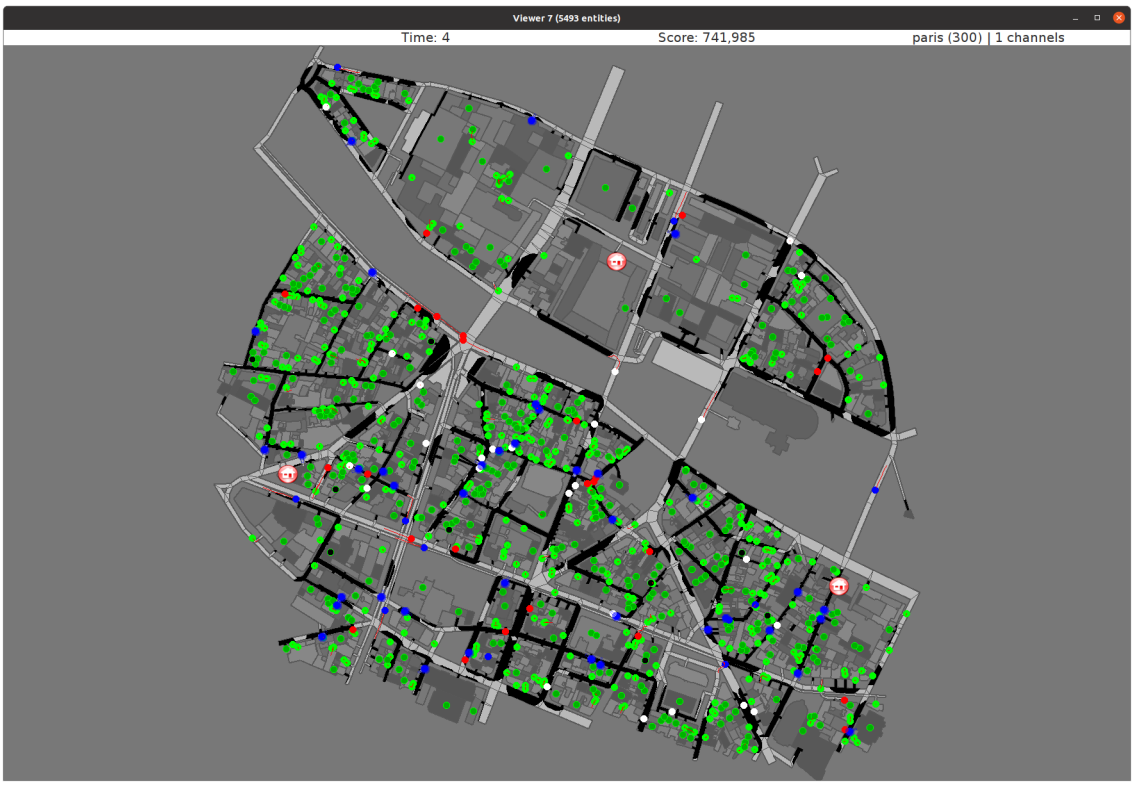


Figura 10: Mapa Paris da RoboCup Rescue Simulation.

Paris é o mapa com distribuição de civis menos uniforme: a região superior apresenta menor densidade em relação à parte inferior e a região central do mapa não possui um refúgio de civis, totalizando apenas 3 deles no mapa, o que dificulta o resgate, visto o grande número de bloqueio nas ruas (partes pretas da figura)

7.1.4 JOAO

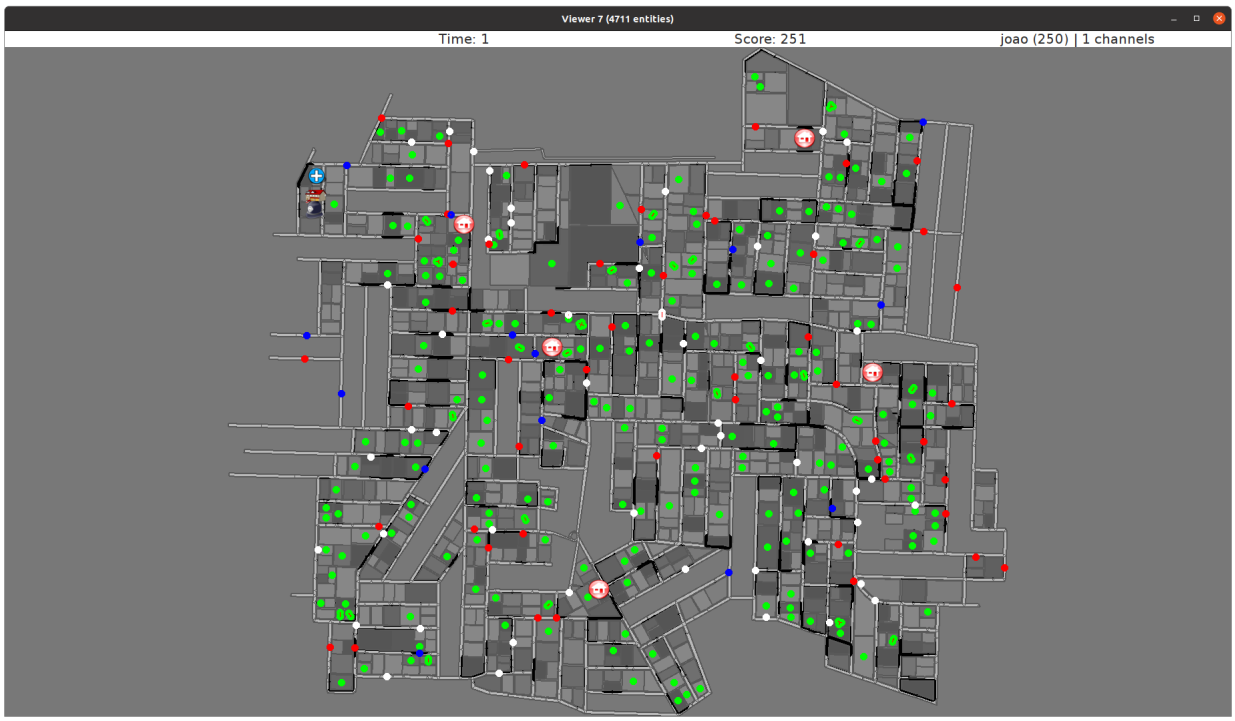


Figura 11: Mapa Joao da RoboCup Rescue Simulation.

Joao é um mapa formado por ruas pequenas e baixa concentração de civis nas bordas do mapa. Ele apresenta 5 refúgios a civis e baixa densidade de feridos.

7.2 APRESENTAÇÃO DE RESULTADOS OBTIDOS

Após as 450 simulações ocorrerem com sucesso, os dados foram salvos em uma árvore de diretórios, onde no final de cada ramo existem logs da simulação da seguinte forma: Time/Mapa/KCluster/Seed/logs. Assim, foi possível desenvolver um outro Script, dessa vez em python, para percorrer todos os caminhos possíveis dessa árvore, e salvar os resultados em um Arquivo CSV.

Para cada uma das simulações coletados algumas métricas como: score final, civis identificados, civis designados, tentativas de resgate, civis carregados pelas ambulâncias, civis resgatados, steps até a identificação, steps de alocação, steps das tentativas, steps de carregamento dos civis e steps dos resgates. Explicadas na Seção 4.3.2.

Ao término de todas as simulações, chegou-se ao volume de 3,5 TB de informações. Foram dias de simulação, distribuídos entre dois computadores. A extração de métricas e o agrupamento de informações relevantes, bem como o armazenamento e a manipulação desses dados não foram tarefas triviais. Comprimir os dados foi essencial para a execução do projeto, mesmo que o tempo de compressão e descompressão também fosse considerável. Pequenas alterações adaptativas foram feitas nos scripts ao longo do tempo visando diminuir o tempo de simulação, conciliando a rapidez de leitura e escrita de SSD's e o armazenamento barato de HDD's. Dessa forma, passou-se a gerar os logs de simulação no primeiro, e, ao término, movê-los para o segundo.

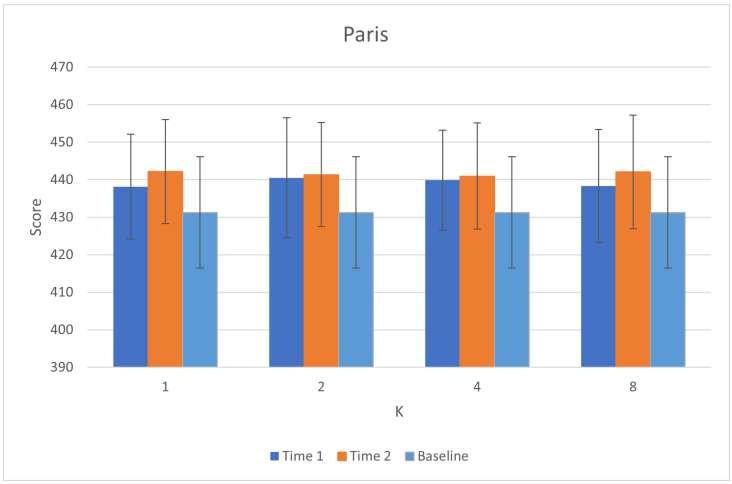
Nessa seção são apresentados os resultados coletados pelo Python Script para avaliação da performance dos times desenvolvidos em relação ao time base fornecido na simulação (Sample). Na seção 7.3 a seguir esses dados são interpretados e analisados.

7.2.1 PONTUAÇÃO COMPARATIVA ENTRE TIMES, COM K'S DIFERENTES, POR MAPA

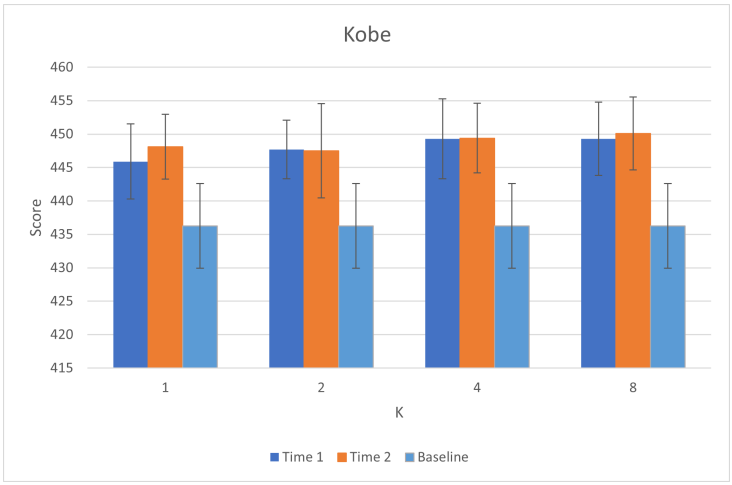
Nesta tabela foi calculada a média e o desvio padrão para cada combinação de Time, KCluster, e Mapa. Na primeira linha de cada bloco é apresentada a média total para o respectivo mapa.

	Time 1		**Time 2**		**Sample**		**Total**	
Mapa	**Média de Score**	**StdDev**	**Média de Score**	**StdDev**	**Média de Score**	**StdDev**	**Média de Score**	**StdDev**
Berlin	**698,31**	**43,56**	**698,12**	**43,91**	**699,24**	**40,99**	**698,55**	**42,84**
K = 1	699,00	42,23	698,04	44,22	699,24	40,99	698,76	42,51
K = 2	697,85	44,41	698,05	44,26	699,24	40,99	698,38	43,25
K = 4	698,22	43,87	697,85	44,43	699,24	40,99	698,44	43,13
K = 8	698,15	43,67	698,53	42,72	699,24	40,99	698,64	42,48
Joao	**153,28**	**8,00**	**153,03**	**8,29**	**153,85**	**9,07**	**153,39**	**8,47**
K = 1	153,05	8,50	153,05	8,29	153,85	9,07	153,32	8,64
K = 2	153,15	7,51	152,95	7,94	153,85	9,07	153,32	8,21
K = 4	153,35	8,12	153,25	8,38	153,85	9,07	153,49	8,54
K = 8	153,55	7,84	152,85	8,51	153,85	9,07	153,42	8,50
Kobe	**448,05**	**5,58**	**448,77**	**5,80**	**436,27**	**6,32**	**444,36**	**8,23**
K = 1	445,89	5,61	448,10	4,86	436,27	6,32	443,42	7,62
K = 2	447,69	4,39	447,49	7,04	436,27	6,32	443,82	8,05
K = 4	449,30	5,97	449,40	5,21	436,27	6,32	444,99	8,50
K = 8	449,30	5,48	450,10	5,46	436,27	6,32	445,22	8,57
Paris	**439,20**	**14,64**	**441,63**	**14,27**	**431,25**	**14,82**	**437,36**	**15,24**
K = 1	438,15	13,97	442,15	13,87	431,25	14,82	437,19	14,92
K = 2	440,45	16,01	441,35	13,84	431,25	14,82	437,69	15,60
K = 4	439,85	13,31	440,95	14,15	431,25	14,82	437,35	14,76
K = 8	438,35	15,01	442,05	15,14	431,25	14,82	437,22	15,65
Berlin2	**761,82**	**30,39**	**761,18**	**30,30**	**755,25**	**28,25**	**759,41**	**29,81**
K = 1	761,09	31,19	756,83	32,09	755,25	28,25	757,72	30,66
K = 2	760,33	30,17	761,21	28,36	755,25	28,25	758,93	29,06
K = 4	764,18	29,18	761,52	30,26	755,25	28,25	760,32	29,48
K = 8	761,67	30,83	765,16	29,79	755,25	28,25	760,69	29,92
Total	**500,13**	**218,00**	**500,55**	**217,73**	**495,17**	**217,33**	**498,62**	**217,70**

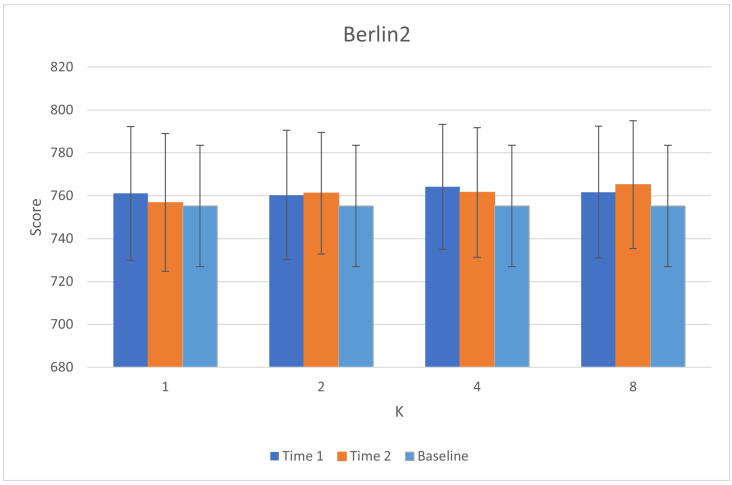
Tabela 1: Scores obtidos nos mapas Berlin, Joao, Kobe, Paris e Berlin2, utilizando diferentes valores de K e testando os diferentes times.



(a) *Paris*



(b) *Kobe*



(c) *Berlin2*

Figura 12: Média de score comparativo entre times, com K's diferentes. Mapas com maiores diferenças nas pontuações

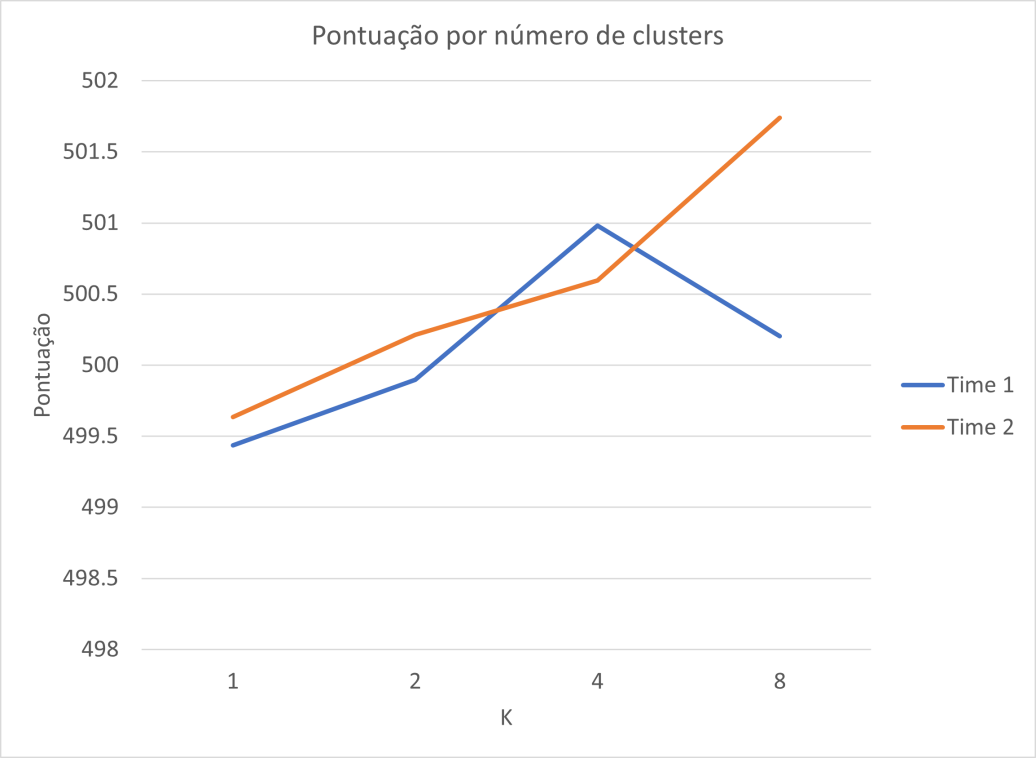


Figura 13: Comparação entre médias de pontuação para cada K em todos os mapas escolhidos.

7.2.2 DESEMPENHO DE MÉTRICAS ENTRE TIMES

	Kobe			Berlin2			Paris			Total		
Parâmetro	**Time 1**	**Time 2**	**Média**	**Time 1**	**Time 2**	**Média**	**Time 1**	**Time 2**	**Média**	**Time 1**	**Time 2**	**Média**
Civis identificados	191,38	202,98	197,18	220,95	235,38	228,16	146,23	163,88	155,05	120,88	130,21	125,54
Civis designados	170,45	164,63	167,54	218,75	223,40	221,08	132,00	140,28	136,14	113,40	115,41	114,40
Tentativas resgate	133,90	121,53	127,71	134,55	127,93	131,24	51,40	47,28	49,34	66,03	60,72	63,37
Civis carregados	79,58	81,25	80,41	87,63	86,15	86,89	21,58	23,93	22,75	39,08	39,05	39,06
Civis resgatados	33,10	73,95	53,53	38,23	60,05	49,14	5,10	15,38	10,24	15,54	30,02	22,78
Score Final	448,05	448,77	448,41	761,82	761,18	761,50	439,20	441,63	440,42	500,13	500,55	500,34

Tabela 2: Avaliação de parâmetros da simulação.

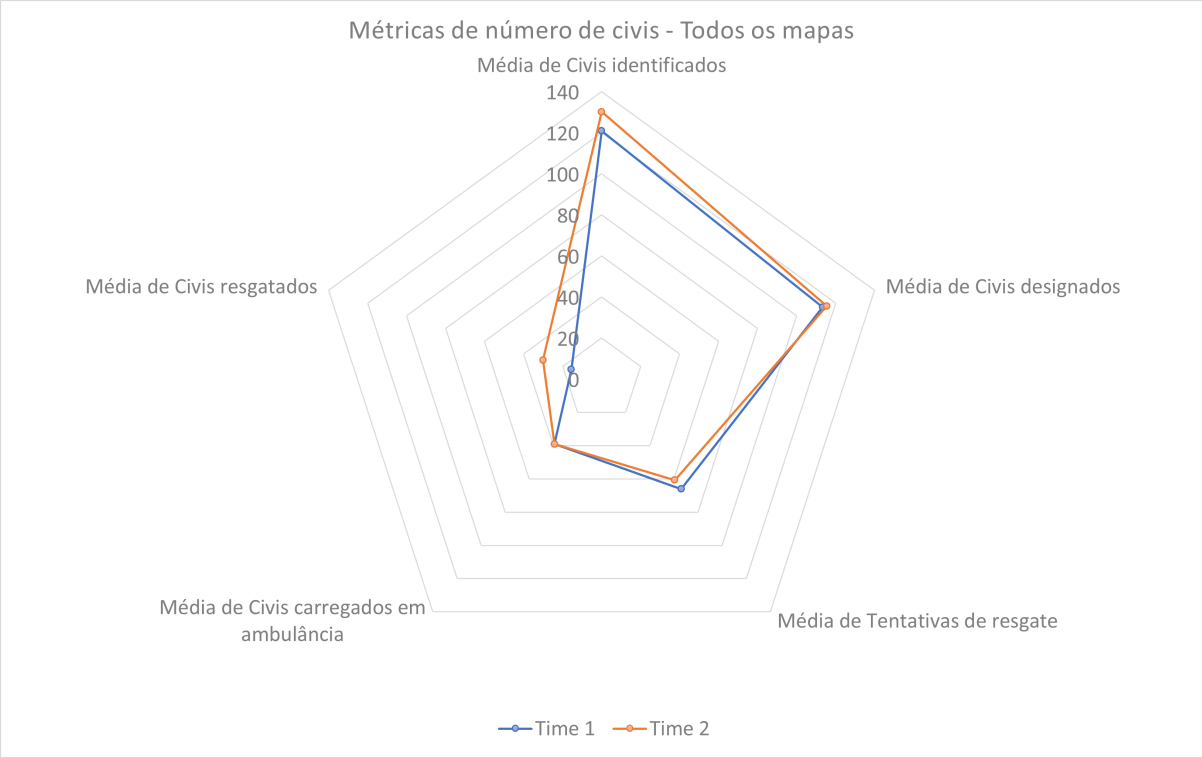


Figura 14: Apresentação visual da média total dos dados da tabela 2, considerando todos os mapas.

	Kobe			Berlin2			Paris			Total		
Parâmetro	**Time 1**	**Time 2**	**Média**	**Time 1**	**Time 2**	**Média**	**Time 1**	**Time 2**	**Média**	**Time 1**	**Time 2**	**Média**
Step identificação	78,27	81,65	79,96	89,94	91,55	90,74	122,96	129,41	126,19	95,90	100,00	97,95
Step de designação	96,26	100,93	98,59	100,15	110,98	105,56	157,01	170,71	163,86	108,44	116,07	112,26
Step das tentativas	114,58	111,10	112,84	110,46	109,40	109,93	186,85	180,34	183,59	132,32	130,25	131,30
Step carregamento	137,19	140,31	138,75	125,98	131,49	128,73	191,85	192,08	191,96	147,23	152,19	149,68
Step dos resgates	140,36	143,44	141,90	132,43	138,24	135,34	252,02	203,62	227,51	174,88	165,04	170,10
Tempo para desenterrar	47,88	48,03	47,95	44,19	45,97	45,08	42,66	45,31	43,98	42,51	47,26	44,85
Score Final	448,05	448,77	448,41	761,82	761,18	761,50	439,20	441,63	440,42	500,13	500,55	500,34

Tabela 3: Avaliação de steps médios da simulação.

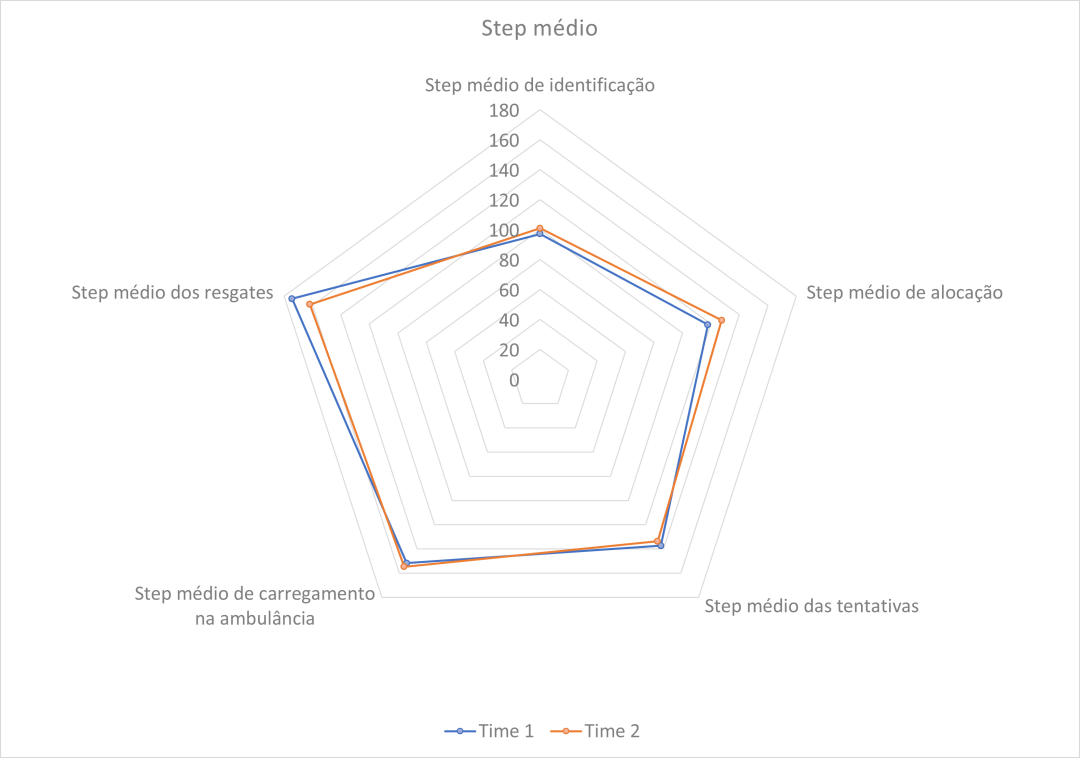


Figura 15: Apresentação visual de dados da tabela 3.

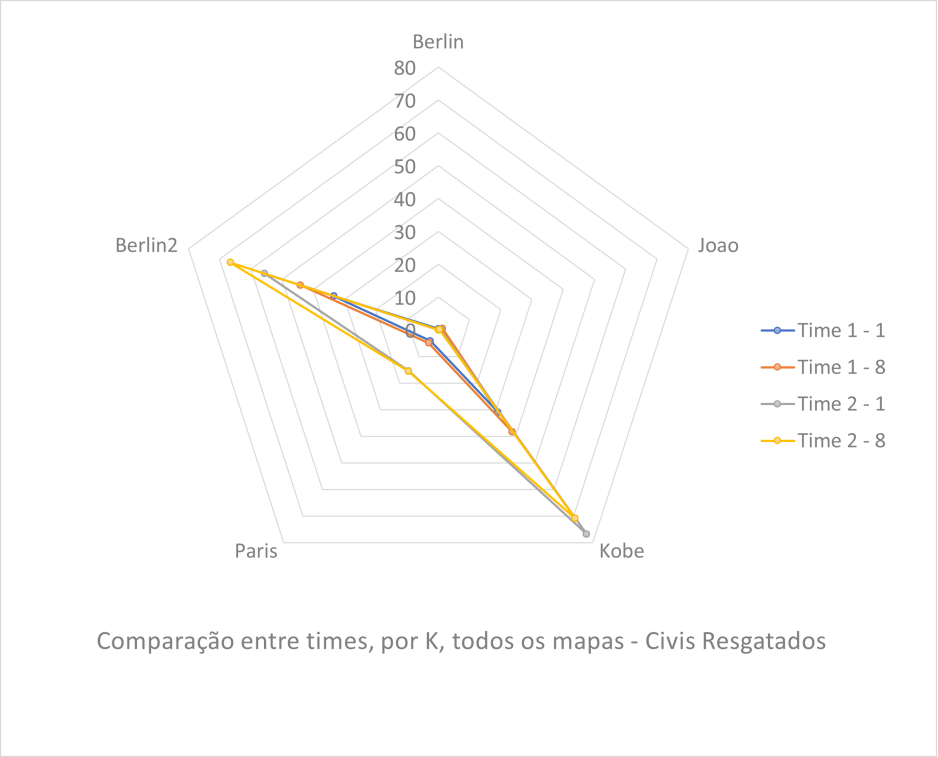


Figura 16: Destaque para quantidade de civis resgatados entre times, e entre K = 1 e K = 8.

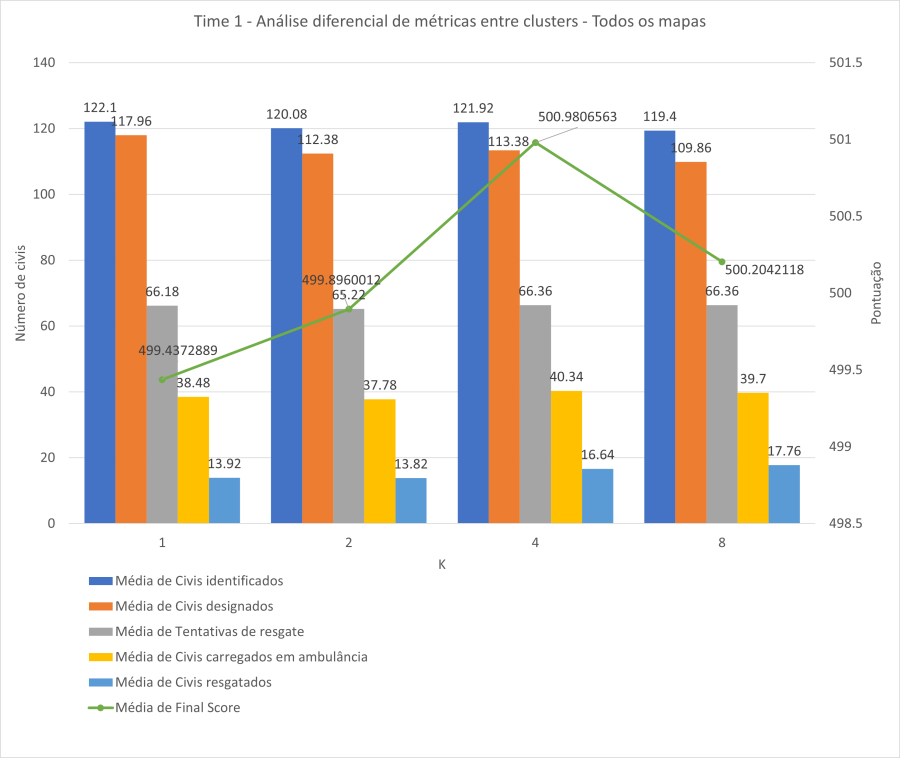


Figura 17: Time 1 - Métricas de número de civis com pontuação, considerando todos os mapas

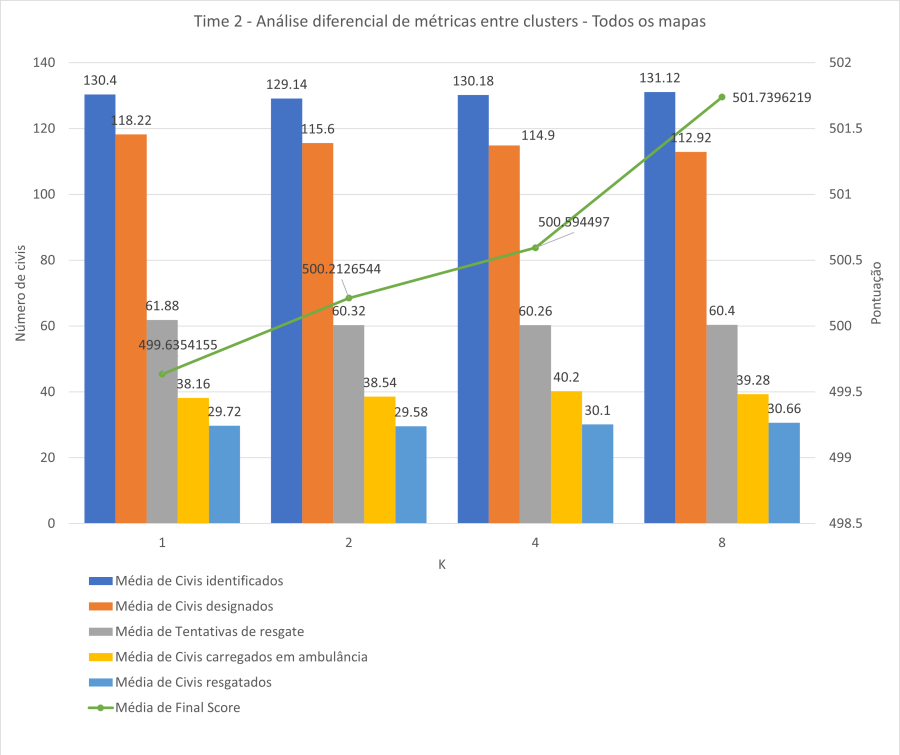


Figura 18: Time 2 - Métricas de número de civis com pontuação, considerando todos os mapas

7.2.3 TESTES DE WILCOXON-MANN-WHITNEY.

Realizamos os testes de Wilcoxon-Mann-Whitney como descrito na seção 4.5. Assim, seguem alguns dados relevantes que obtiveram o p-value abaixo de 0.05, o que demonstra uma diferença de distribuição entre os dados testados. Como o valor de p é baixo, podemos comparar as médias para dizer o que provavelmente é maior ou menor. Outros valores para pontuação, número de civis resgatados, número de civis mortos não tiveram diferença significante.

Parâmetro	Mapa	Time a	Média	Time b	Média	statistic	p-value
Civis mortos	**Kobe**	Sample	64,40	Time 1	51,92	4,075	0,000046
Civis mortos	**Kobe**	Sample	64,40	Time 2	52,27	3,881	0,000104
Step das mortes de civis	**Kobe**	Sample	137,76	Time 1	127,01	2,692	0,007099
Step das mortes de civis	**Kobe**	Sample	137,76	Time 2	127,13	2,765	0,005694
Civis identificados	**Berlin2**	Time 1	220,95	Time 2	235,38	-1,968	0,049091
Civis identificados	**Kobe**	Time 1	191,38	Time 2	202,97	-3,026	0,002476
Civis identificados	**Paris**	Time 1	146,22	Time 2	163,88	-3,594	0,000326
Civis resgatados	**Berlin2**	Time 1	38,23	Time 2	60,05	-5,398	0,000000
Civis resgatados	**Kobe**	Time 1	33,10	Time 2	73,95	-7,332	0,000000
Civis resgatados	**Paris**	Time 1	5,10	Time 2	15,38	-7,289	0,000000
Step de resgates de civis	**Berlin2**	Time 1	132,43	Time 2	138,24	-3,315	0,000917
Step de resgates de civis	**Paris**	Time 1	252,02	Time 2	203,62	6,795	0,000000

Tabela 4: Teste de Wilcoxon para diferentes times no mesmo mapa.

Parâmetro	Time 1	Time 2	statistic	p-value
Step de designação	108,44	116,07	-3,278	0,001045
Step de identificação	95,9	100	-2,156	0,031060
Civis resgatados	15,54	30,02	-2,769	0,005628
Civis em resgate	66,03	60,72	2,841	0,004498

Tabela 5: Teste de Wilcoxon para o Time 1 e Time 2.

Parâmetro	Mapa	K	Média	K	Média	statistic	p-value
Civis mortos	**Kobe**	1	57,00	4	51,05	2,683	0,007290
Civis mortos	**Kobe**	1	57,00	8	50,65	2,733	0,006280
Score Final	**Kobe**	1	443,42	4	449,35	-2,693	0,007077
Score Final	**Kobe**	1	443,42	8	449,70	-2,654	0,007964
Civis resgatados	**Berlin2**	1	44,50	8	55,30	-2,380	0,017293

Tabela 6: Teste de Wilcoxon para diferentes K's.

7.3 ANÁLISE DOS RESULTADOS

7.3.1 PONTUAÇÃO

Na tabela 1 encontram-se as médias de pontuação de todas as Seeds, para cada time, assim como o respectivo desvio padrão. Pode-se observar que, para dois mapas, Berlin e Joao, o desempenho dos times desenvolvidos pelo grupo foi muito próximo ao time Sample, tomado como baseline, sem diferença significativa. Entretanto, a diferença é bem substancial para os mapas Kobe e Paris, independente do número de clusters. Chegou-se à conclusão que o baixo desempenho nos dois primeiros se deve ao elevado número de bloqueios que são gerados nestes mapas, unidos a seus tamanhos consideráveis. A descoberta desta informação fomentou a edição e adição de um novo mapa às simulações, Berlin2. Em Berlin, o time Sample foi ligeiramente melhor que os Time 1 e 2, enquanto em Berlin2, versão sem os bloqueios das ruas, os resultados se inverteram. Essa constatação trouxe o conhecimento de que a clusterização do ambiente só é aproveitada até certo nível de mobilidade pelo mapa.

Além disso, percebe-se que a centralização e organização de comandos traz um ganho muito mais significativo do que a clusterização do mapa. Isso pode ser visto ao se comparar K = 1 entre times, e, subsequentemente, K = 8. Entretanto, não se pode descartar a diferença trazida pelos diferentes valores de K, como evidenciado pelo gráfico da figura 13. O gráfico nos diz que o time 2 aproveita mais a clusterização, tendo maiores ganhos com o aumento do número de K's. Apesar da diferença gráfica, a diferença de menos de 2 pontos na Pontuação não se mostrou significativa estatisticamente. Essa maior relação entre K e pontuação se evidencia nos mapas com menos barreiras nas ruas, como é o caso de Berlin2 e Kobe, o que mostra que não só a centralização dos comandos fazem diferença em um mapa mais livre (ao se comparar Time 1, Time 2 e Sample com K = 1), como o aumento do K também beneficia mais os Times 1 e 2 nesses mapas (ao se comparar K = 1 com K = 4 ou 8). Essa diferença se mostra estatisticamente relevante na Tabela 6.

7.3.2 MÉTRICAS

De todas as métricas coletadas na seção 4.3.2, apenas o Score, o número de civis mortos e o Step dos mortos compreendem o time de baseline. O motivo pelo qual não foi possível coletar o resto das métricas para o Sample foi principalmente o tempo que seria necessário para ajustar o código do time de exemplo para escrever no log as informações. Esses dados foram coletados pela central de ambulâncias nos times desenvolvidos pelo grupo,

código que não é executado no time exemplo por ter todas as ambulâncias trabalhando de forma autônoma. Também, escolheu-se mostrar o resultado somente nos mapas em que os times tiveram melhora em relação ao baseline, para que eventuais diferenças fossem mais acentuadas para fins de melhor compreensão

Em relação a diferença de performance entre os times 1 e 2, que pode ser visualizada nas figuras 17 e 18 juntamente com a tabela 5, vê-se uma diferença substancial na quantidade de civis resgatados para qualquer K escolhido. Também é possível ver, de forma ligeiramente reduzida, uma grande diferença na quantidade de civis identificados. No entanto, para as métricas de civis alocados e tentativas de resgate, observa-se que, para a primeira, existe pouca mudança entre times. Já para a segunda, curiosamente existem mais tentativas de resgate para o primeiro time que para o segundo, embora o resultado de civis salvos seja melhor para este último. Percebe-se que o time 2 parece ser mais eficiente em relação à alocação de alvos para ambulância, atingindo melhores resultados com menos tentativas de resgate.

Uma boa explicação para esse comportamento é começar a resgatar os civis tardiamente, mais no fim da simulação, e isso se pode ver no gráfico apresentado na figura 15, em que a média do step de tentativas de resgate é um pouco maior que o time 2. A consequência do resgate tardio pode ser a morte do civil durante as tentativas de resgate ou mesmo durante o transporte até o refúgio. Outra explicação advém da visualização da simulação, na qual é possível perceber, de forma genérica, que o time 1 passa a se comportar de forma mais autônoma na parte final da simulação. Isso acaba implicando a aglomeração de várias ambulâncias fazendo a mesma tarefa e o movimento sem restrições pelo mapa, aumentando o tempo de locomoção. Na prática, isso pode trazer ajuda de ambulâncias de outros clusters, mas acaba atrasando o resgate por conta do tempo de locomoção, tanto ao irem ajudar, quanto ao voltarem para o cluster de origem ao serem alocadas.

Ainda nas mesmas figuras 17 e 18, também se observa a ligeira melhora da métrica Civis Resgatados, com o aumento dos K's, para ambos os times. Além essa métrica, na tabela 6 é possível ver que a diferença foi estatisticamente relevante para Civis mortos, Score Final e Civis resgatados para os mapas Kobe e Berlin2. Essa diferença foi sempre positiva com o aumento de K, e nem uma vez foi negativa ao se fazer o teste estatístico.

Por último, na Tabela 4 é possível ver que para alguns mapas, o Time 2 supera o Time 1 nas métricas de Civis Identificados, Civis Resgatados. Ambos os Times 1 e 2 superam o Sample no número de mortes e tem o tempo médio de morte menor. Todas

essas análises se mostraram estatisticamente relevantes. Em relação ao step médio de resgate, entende-se que é difícil fazer uma conclusão estatisticamente relevante a respeito pois em mapas específicos um time desempenhou melhor que outro. Entretanto, pode-se entender que a média favorecer o time 2, sendo este o time que também mais resgata muito mais civis, mantendo a média de step de resgate próxima.

Dessa forma, reiterando-se o que foi constatado no item anterior, percebe-se que a centralização e organização de comandos traz um ganho muito mais significativo do que a clusterização do mapa, apesar da clusterização ser relevante nos mapas considerados mais livres, com menos bloqueios nas ruas, como Kobe e Berlin2.

8 CONSIDERAÇÕES FINAIS

Um dos principais questionamentos que ficam é: Se o número de civis resgatados pelo time 2 é quase o dobro do time 1, porque a pontuação não reflete isso? Uma possível resposta está na saúde dos civis resgatados, já que a pontuação final depende também deste valor. Quanto mais tempo o civil passa soterrado, menor seu atributo de saúde. O fato de o time 2 ter uma melhor distribuição de ambulâncias acaba significando ter um menor número destas atuando no salvamento de cada alvo, e isso, por sua vez, leva-o a ter menos pontos de vida no momento do resgate.

A análise das métricas nos mostra que, de alguma forma, os times 1 e 2 são complementares, melhorando em um aspecto e piorando em outro de forma que, no fim, o desempenho é semelhante. Talvez um novo time com uma implementação que vise a aproveitar o melhor aspecto de cada um seja mais eficiente que os dois.

Em questão de clusterização, acredita-se que pra cada mapa um valor diferente de k seja ideal, além de também para cada time. Uma clusterização dinâmica, feita após o reconhecimento do mapa, talvez seja uma melhora considerável frente a um k predeterminado.

8.1 CONCLUSÕES DO PROJETO DE FORMATURA

Os objetivos do projeto, apresentados na seção 1.1, como otimizar a colaboração de agentes autônomos no planejamento e na execução de resgates em meio a desastres naturais na liga de simulação de agentes RoboCup Rescue foram atingidos, uma vez que o algoritmo implementado apresentou impactos positivos no desempenho da simulação. Esse fato foi verificado por meio da avaliação estatística utilizando o Teste Wilcoxon-Mann-Whitney, testando os resultados da setorização inteligente do ambiente, em que cada um dos setores é definido pela unidade central de controle das ambulâncias da simulação.

O algoritmo também foi avaliado em diferentes mapas da simulação, buscando minimizar vieses geográficos que podem existir por conta de particularidades de certos mapas, tais como a maior concentração de civis em áreas litorâneas do mapa ou regiões com baixa densidade de prédios.

O trabalho compreendeu uma ampla parte de conhecimentos adquiridos ao longo do curso dos alunos envolvidos, dos quais podem-se citar Arquitetura de Computadores, Estatística, Programação Orientada a Objetos, Algoritmos e Estrutura de Dados e Inteligência Artificial como sendo algumas das mais utilizadas dentre tantas outras essenciais, tanto para o projeto em questão quanto para a formação como Engenheiros da Computação.

8.2 CONTRIBUIÇÕES

Por meio deste trabalho o grupo conseguiu atingir uma melhora nos resultados da simulação RoboCup Rescue por meio da implantação centralizada do algoritmo de clusterização K-Means, conforme apresentado no capítulo 7. Pode-se verificar que foi possível aumentar o número de sobreviventes no desastre simulado e reduzir o tempo para localizar e resgatar civis.

Dessa forma, apresenta-se uma evolução na estratégia de resgate de civis da RoboCup Rescue Simulation, que está inserida no contexto de longo prazo da RoboCup de criar um time de humanóides completamente autônomo que seja capaz de vencer o último ganhador da Copa do Mundo até o meio do século XXI.

Além disso, neste trabalho foi implementada uma interface visual para compreensão do funcionamento dos clusters, contribuindo também para o aprendizado de alunos ou para facilitar o processo de correções de erros de outros desenvolvedores que trabalhem no projeto.

8.3 PERSPECTIVAS DE CONTINUIDADE

Em um trabalho futuro, existem possibilidades de evolução e a seguir listamos três delas:

- **Implementação de outros algoritmos**

  Pode-se desenvolver outros algoritmos de clusterização como o DBSCAN para ava-

liação do impacto em métricas como o tempo médio de salvamento dos civis. O algoritmo DBSCAN (Density-Based Spatial Clustering of Applications with Noise) busca por grupos definidos como regiões com alta densidade de objetos, separados por regiões de baixa densidade. Uma das principais vantagens desse algoritmo advém do fato de não ser necessário informar previamente o número desejado de grupos. Dessa forma, na avaliação dos resultados obtidos na simulação não é necessário executar o algoritmo com diferentes valores de K.

- **Avaliação de resultados em outros mapas**

  Pode-se utilizar outros mapas na avaliação de resultados obtidos, uma vez que particulares espaciais da distribuição dos prédios podem ter impactos significativos na distribuição de ambulâncias. Uma distribuição uniforme tende a resultar em uma alocação de ambulâncias similar entre as áreas do mapa, fato que não ocorre no caso de uma área do mapa concentrar um grande número de prédios, como em uma região litorânea com grande concentração de edifícios em frente à orla da praia.

- **Comparação de resultados com algoritmos não centralizados**

  Pode-se comparar os resultados obtidos com a clusterização na central de ambulâncias em relação à clusterização não centralizada que ocorre iterativamente em cada instante de execução da simulação. Também, é interessante comparar o desempenho de algoritmos não centralizados melhorados em relação ao sample, com melhor comunicação entre agentes ao invés de entre agente e central

REFERÊNCIAS

1 TEIXEIRA, J. de F. *Inteligência artificial*. Paulus Editora, 2014. (Como ler filosofia). ISBN 9788534936842. Disponível em: ⟨https://books.google.com.br/books?id=79q5DAAAQBAJ⟩.

2 ROBOCUP. A brief history of robocup. RoboCup. Disponível em: ⟨https://www.robocup.org/a_brief_history_of_robocup⟩.

3 BRANISSO, L. B. Sistema multiagente para controle de veículos autônomos. Künstliche Intelligenz, São Carlos, SP, Brasil, 2014.

4 NETO, J. P. B. Modelagem de um sistema multiagente para aplicação em ambientes industriais. Universidade Federal de Campina Grande, Campina Grande, PB, Brasil, 2007.

5 COSTA, A. H. R. Transferência de conhecimento no aprendizado por reforço em sistemas multiagentes. Escola Politécnica da Universidade de São Paulo (USP), São Paulo, SP, Brasil, 2015.

6 SHEH, R.; SCHWERTFEGER, S.; VISSER, A. 16 years of robocup rescue. Curtin University, Amsterdam, The Netherlands, 2016.

7 JAMES, G. et al. *An introduction to statistical learning*. Springer New York Heidelberg Dordrecht London, 2013. (Ensino). ISBN 9781461471370. Disponível em: ⟨https://www.ime.unicamp.br/~dias/Intoduction%20to%20Statistical%20Learning.pdf⟩.

8 WOOLDRIDGE, M. J. *Introduction to Multiagent Systems*. New York, NY, USA: John Wiley & Sons, Inc., 2009.

9 BAJO, J. et al. *Highlights of Practical Applications of Agents, Multi-Agent Systems, and Sustainability: The PAAMS Collection: International Workshops of PAAMS 2015, Salamanca, Spain, June 3-4, 2015. Proceedings*. Springer International Publishing, 2015. (Communications in Computer and Information Science). ISBN 9783319190334. Disponível em: ⟨https://books.google.com.br/books?id=9CxACQAAQBAJ⟩.

10 BAKER, B. et al. Emergent tool use from multi-agent autocurricula. 2019.

11 SILVA, A.; NARDIN, L.; SICHMAN, J. Um método baseado em particionamento para exploração de ambientes de desastre. In: . [S.l.: s.n.], 2012.

12 EASTERN KENTUCKY UNIVERSITY. *Infographic on the Benefits and Challenges of Using AI for Emergency Management*. 2018. Disponível em: ⟨https://safetymanagement.eku.edu/blog/the-benefits-challenges-of-using-artificial-intelligence-for-emergency-management/⟩.

13 PONOMAREV, S.; VORONKOV, A. E. Multi-agent systems and decentralized artificial superintelligence. 2017.

14 SICHMAN, J. S.; GARCIA, A. C. B. Agentes e multiagentes. Editora Manole LTDA., São Paulo, SP, Brasil, 2005.

15 BOND, A. H.; GASSER, L. A survey of distributed artificial intelligence. 1988.

16 MICHEL, J. F. F.; DROGOUL, A. Multi-agent systems and simulation: a survey from the agents community's perspective. 2001.

17 HOLLAND, O. Multiagent systems: Lessons from social insects and collective robotics. 1996.

18 STONE, P.; VELOSO, M. Multiagent systems: A survey from a machine learning perspective. 2000.

19 SILVA, C.; RIBEIRO, B. *Aprendizagem Computacional em Engenharia*. Imprensa Da Universidade de Coimbra / Coimbra Univ, 2018. (Ensino). ISBN 9789892615073. Disponível em: ⟨https://books.google.com.br/books?id=oFhQDwAAQBAJ⟩.

20 JAMES, G. et al. *An Introduction to Statistical Learning: with Applications in R*. Springer New York, 2014. (Springer Texts in Statistics). ISBN 9781461471370. Disponível em: ⟨https://books.google.com.br/books?id=at1bmAEACAAJ⟩.

21 BOSLAUGH, S.; WATTERS, P. *Statistics in a Nutshell: A Desktop Quick Reference*. O'Reilly Media, 2008. (In a Nutshell (O'Reilly)). ISBN 9781449397814. Disponível em: ⟨https://books.google.com.br/books?id=ZnhgO65Pyl4C⟩.

22 HOLLANDER, M.; WOLFE, D.; CHICKEN, E. *Nonparametric Statistical Methods*. Wiley, 2013. (Wiley Series in Probability and Statistics). ISBN 9781118553299. Disponível em: ⟨https://books.google.com.br/books?id=Y5s3AgAAQBAJ⟩.

23 NARDIN, L. G. Uma arquitetura de apoio à interoperabilidade de modelos de reputação de agentes. Escola Politécnica da Universidade de São Paulo (USP), São Paulo, SP, Brasil, 2009.

24 LEAGUE, R. R. Agent simulation competition. 2020. Disponível em: ⟨https://rescuesim.robocup.org/competitions/agent-simulation-competition/⟩.

25 TAHERIAN, M.; NARDIN, L. G. Visual debugger. 2016. Disponível em: ⟨https://github.com/roborescue/visual-debugger⟩.

26 GOHARDANI, P. D.; NIKOO, S. M. P. A. E. J.; TAHERIAN, M. Robocup rescue 2016 - infrastructure team description - mrl (iran). 2016. Disponível em: ⟨PooyaDeldarGohardani,SiavashMehrabi,PeymanArdestani,ErfanJazebNikoo, MahdiTaherian⟩.

27 BACH, J. *Bash Scripting: Learning the Bash Shell, 1st Edition*. Independently Published, 2020. ISBN 9798685100719. Disponível em: ⟨https://books.google.com.br/books?id=pT4GzgEACAAJ⟩.

28 PIERSON, L.; PORWAY, J. *Data Science For Dummies*. Wiley, 2017. (–For dummies). ISBN 9781119327639. Disponível em: ⟨https://books.google.com.br/books?id=Uz5GDgAAQBAJ⟩.

APÊNDICE A – BASH SCRIPT

```
#!/bin/bash
source ./config.sh
TEAM='rcrs-adf-poli'

function waitFor {
    SLEEP_TIME=1
    FREQUENCY=1
    if [ ! -z "$3" ]; then
        FREQUENCY=$3
    fi
    if [ ! -z "$4" ]; then
        SLEEP_TIME=$4
    fi
    F=$FREQUENCY
    while [[ ! -e $1 ]]; do
        if (( --F == 0 )); then
            F=$FREQUENCY
        fi
        sleep $SLEEP_TIME
    done
    while [ -z "`grep \"$2\" \"$1\"`" ]; do
        if (( --F == 0 )); then
            F=$FREQUENCY
        fi
        sleep $SLEEP_TIME
    done
}
```

```
function start_server {
  gnome-terminal --title "rcrs-server" -- /bin/sh -c 'cd '$SERVERDIR';
  ↪  mkdir -p '$2'; bash start-comprun.sh -c '$MAPSDIR'/'$1'/config -m
  ↪  '$MAPSDIR'/'$1'/map -l '$2' -g'
}

function start_agent {
  gnome-terminal --title "agent" -- /bin/sh -c 'cd ..; sh launch-poli.sh'
}

function start_visual_debugger {
  gnome-terminal --title "visual-debugger" -- /bin/sh -c 'cd
  ↪  ../../../visual-debugger/; ./gradlew launch'
}

for SEED in ${SEEDS[*]};
do


  for K in ${CLUSTERS[*]};
  do

    #mudar a linha com o numero de clusters
    cluster_line="/*numero de clusters*/  public static final int
    ↪  nClusters = $K;"
    awk -v c="/*numero" -v nc="$cluster_line" '$1 == c { $0 = nc } 1'
    ↪  PoliConstants.java >../src/poli/PoliConstants.java
    #limpar o agente e recompila-lo
    gnome-terminal --title "agent-build" --disable-factory -- /bin/sh -c
    ↪  'cd ..; ./gradlew clean; ./gradlew build'

    for MAP in ${MAPS[*]};
    do
      #mudar a linha com a seed da simulação
      seed_line="random.seed: $SEED"
```

```
        awk -v c="random.seed:" -v nc="$seed_line" '$1 == c { $0 = nc } 1'
        ↪  common.cfg >"$SERVERDIR"/"$MAPSDIR"/"$MAP"/config/common.cfg
        LOGDIRATUAL2="$LOGDIR2/$TEAM/$MAP/$K"
          LOGDIRATUAL="$LOGDIR/$TEAM/$MAP/$K/$SEED"
        echo "Launching simulation: $MAP -> k = $K and seed = $SEED and
        ↪  path = $LOGDIRATUAL2/$SEED"
        if [ -d "$LOGDIRATUAL2/$SEED/logs" ]; then
          echo "Skipping..."
          continue
        fi
        #iniciar o servidor
        start_server $MAP $LOGDIRATUAL
        sleep 5
        #iniciar agente
              if [ "$DEBUG" = true ]; then
          start_visual_debugger
          sleep 5
        fi
        start_agent
        #esperar o fim do simulador
        waitFor $LOGDIRATUAL/kernel.log "Kernel has shut down" 30
        #fechar o agente
        pkill -f 'cd ..; sh launch-poli.sh'
              if [ "$DEBUG" = true ]; then
          pkill -f 'cd ~/Documents/tcc/src/visual-debugger/; ./gradlew
          ↪  launch'
        fi
        gnome-terminal --title "copy" --disable-factory -- /bin/sh -c 'cp
        ↪  -r ../logs '$LOGDIRATUAL'/logs'
        grep "Score:" $LOGDIRATUAL/kernel.log >$LOGDIRATUAL/score.log
        gnome-terminal --title "move" --disable-factory -- /bin/sh -c
        ↪  'mkdir -p '$LOGDIRATUAL2'; mv '$LOGDIRATUAL' '$LOGDIRATUAL2';'
      done ;

    done ;
done ;
```

```
echo "finished script"
```

Bash Script criado para automatizar a captura de resultados das simulações.

```
export SERVERLOG='../../../rcrs-server/boot/logs' # Caminho do log da
↪  simulação
export SERVERDIR='../../../rcrs-server/boot' # Caminho do diretorio boot
↪  do servidor
export MAPSDIR='../maps/gustavo' # Caminho dos mapas em relação ao
↪  caminho do SERVERDIR
export LOGDIR='/home/gabriel/LOGS' # Caminho de onde salvar todos os
↪  logs da simulação
export DEBUG=true # Iniciar ou não debug viewer
export LOGDIR2='/media/gabriel/Appa/LOGS' # Caminho de onde salvar todos
↪  os logs da simulação

#MAPS=(berlin joao kobe paris)
export MAPS=(berlin2 paris kobe berlin joao)
#CLUSTERS=(1 2 4 8)
export CLUSTERS=(8 4 2 1)
#SEEDS=(90 198 1234)
export SEEDS=(90 198 1234 5531 8051)
```

Arquivo config.sh utilizado com o Bash Script anterior.

APÊNDICE B – SEED

```
include types.cfg

# Random seed for all components
random.seed: 3

kernel.host: localhost
kernel.port: 7000
senario.human.random-id: true
```

Código do arquivo common.cfg do mapa Kobe contendo uma seed.