ENGENHARIA DE SOFTWARE BASEADA EM BUSCA (SBSE)

Equipe Marvin -

- Bruna Mariana F. de Souza
- Débora da C. Medeiros
- Valber C. Tavares

ICOMP
Instituto de Computação

EVOSUITE 

Estudo de Caso

Repositório	Commits	Contribuidores	Branches	Linguagem Principal	Licença
MyRobotLab	14,391	24	42	Java	Apache License 2.0

ICOMP
Instituto de Computação

EVOSUITE 

Configuração do Ambiente de Desenvolvimento:

- Sistema Operacional: Linux
- JDK (Java Development Kit): java 11.0.22
- Apache Maven: Apache Maven 3.6.3
- EvoSuite: Versão 1.2.1

Atualizações da versão EvoSuite 1.2.1 -

- Suporte Java 9 para DSE
- Configuração atualizada do Docker
- Suporte para sa´ıda JUnit 5
- Refatoração do código relacionado a genéricos
- Suporte Java 9 para plug-in Eclipse
- Refatoração e reformatação gerais
- Correções de bugs para vários travamentos e problemas menores

ICOMP
Instituto de Computação

EVOSUITE

Desafios com a versão EvoSuite 1.2.1

```
for I in {1..10}; do sudo java -jar ~/myrobotlab/evosuite/master
   /target/evosuite-master-1.2.1-SNAPSHOT.jar -target ./target/
   classes -Doutput_variables=TARGET_CLASS,criterion,Coverage,
   Total_Goals,Covered_Goals,Size,Length,MutationScore ; done
```

ICOMP
Instituto de Computação

EVOSUITE

Desafios com a versão EvoSuite 1.2.1

```
#!/bin/bash

# Diret rio onde est o localizadas as classes de teste
TEST_DIR="./evosuite-tests"

# Comando a ser executado para cada classe de teste
COMMAND="sudo␣java␣--add-opens␣java.desktop/java.awt=ALL-UNNAMED
   ␣-cp␣.:\$HOME/dados_evosuite_myrobotlab/myrobotlab/target/
   classes:\$HOME/dados_evosuite_myrobotlab/myrobotlab/evosuite-
   tests-compiled:\$HOME/dados_evosuite_myrobotlab/myrobotlab/
   evosuite/master/target/evosuite-master-1.2.1-SNAPSHOT.jar:\
   $HOME/Tutorial_Stack/target/dependency/junit-4.12.jar␣org.
   junit.runner.JUnitCore"

# Arquivo para salvar a sa da dos comandos
LOG_FILE="test_results.log"
# Arquivo para salvar os nomes das classes com falhas
FAILURE_FILE="failures.log"

# Cria ou limpa os arquivos de log
> "\$LOG_FILE"
> "\$FAILURE_FILE"

# Vari vel para contar os testes OK
tests_ok=0
# Vari vel para contar o total de testes
total_tests=0

# Loop recursivo para percorrer todas as classes de teste
for file in \$(find "\$TEST_DIR" -type f -name '*.java'); do
    # Verifica se o nome do arquivo cont m a palavra "
       scaffolding"
    if [[ "\$file" == *scaffolding* ]]; then
        continue  # Pula o arquivo se contiver "scaffolding" no
           nome
    fi

    # Extrai o caminho relativo da classe de teste
    relative_path="\${file#\$TEST_DIR/}"
    # Remove a extens o .java do nome do arquivo
    class_name="\${relative_path%.java}"
    # Substitui '/' por '.' para obter o nome do pacote completo
    class_name="\${class_name//\//.}"

    # Executa o comando para a classe de teste
    \$COMMAND "\$class_name" >> "\$LOG_FILE" 2>&1

    # Verifica se o teste foi bem-sucedido (exit code 0)
    if [ \$? -eq 0 ]; then
        ((tests_ok++))
    else
        # Salva o nome da classe com falha no arquivo
        echo "\$class_name" >> "\$FAILURE_FILE"
    fi
done

# Extrai a quantidade de testes do arquivo de log e incrementa
   no total de testes
total_tests=\$(grep -a -o 'OK␣([0-9]\+␣test[s]*' "\$LOG_FILE" |
   sed 's/OK␣(\([0-9]\+\)␣test[s]*.*/\1/' | paste -sd+ - | bc)

echo "Quantidade␣de␣classes␣OK:␣\$tests_ok"
echo "Classes␣com␣falhas:␣\$(cat␣\$FAILURE_FILE)"
echo "Total␣de␣testes␣executados:␣\$total_tests"
```

ICOMP
Instituto de Computação

EVOSUITE - Resultados

Testes antes do EvoSuite:

- Número total de classes: 164
- Total de testes executados: 303
- Falhas: 0

Testes após o EvoSuite:

- Número total de classes: 212
- Total de testes executados: 2707
- Falhas: 0

ICOMP
Instituto de Computação

EVOSUITE - Resultados

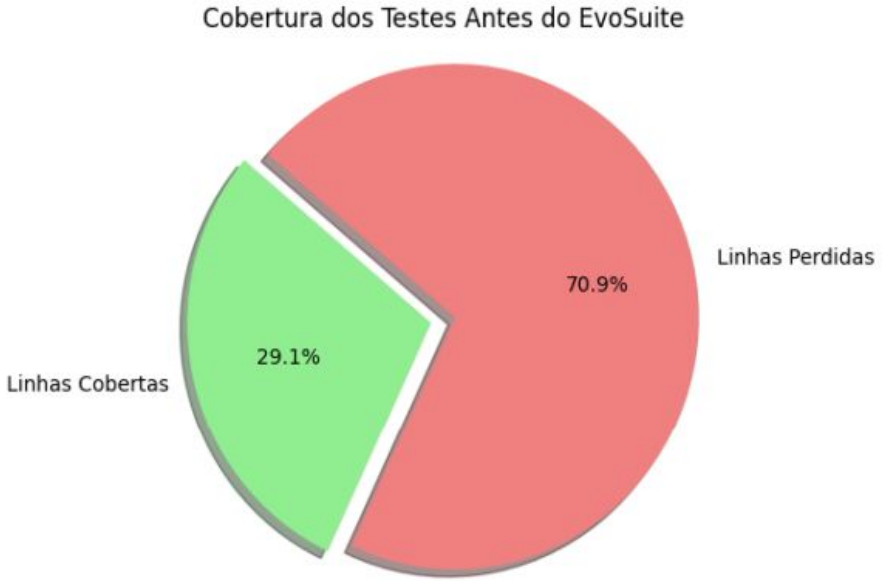


Figure 1. Cobertura antes da aplicação do EvoSuite

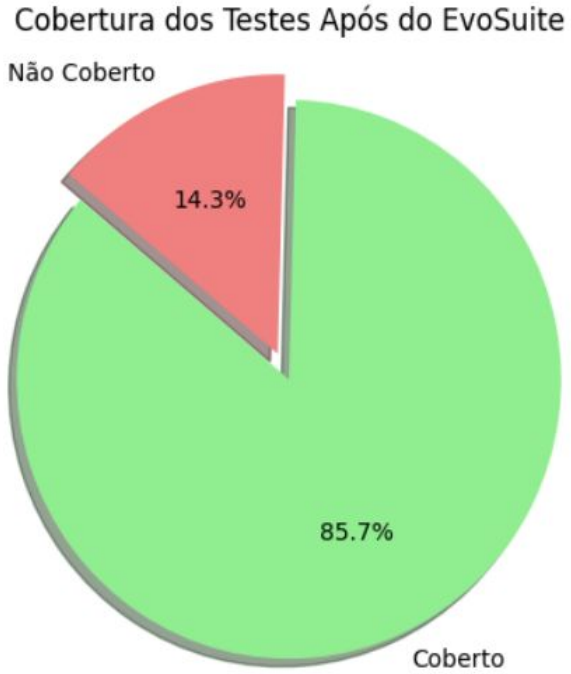


Figure 2. Cobertura antes da aplicação do EvoSuite

ICOMP
Instituto de Computação

EVOSUITE - Resultados

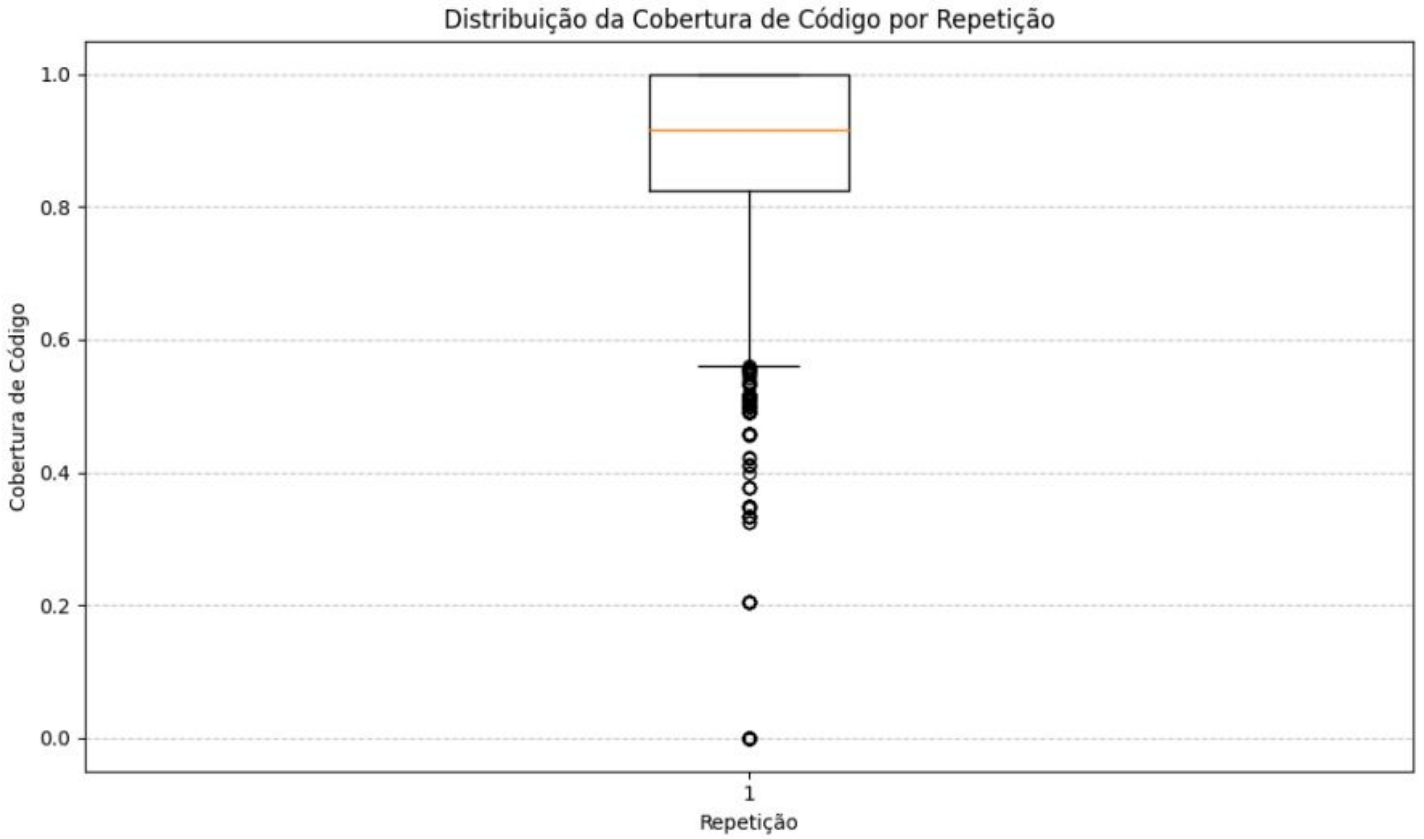
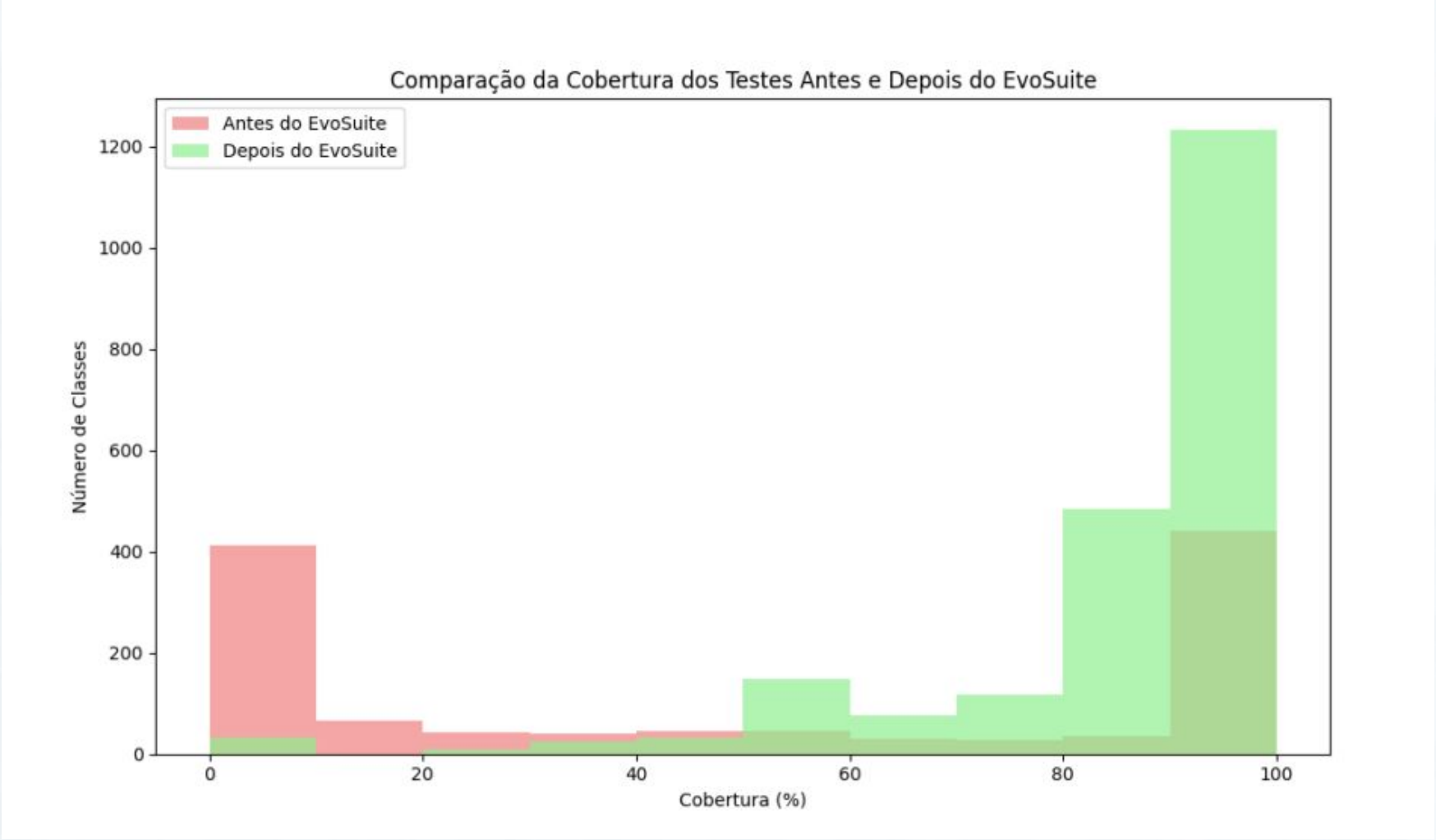


Figure 3. Cobertura por repetição

ICOMP
Instituto de Computação

EVOSUITE - Resultados



ICOMP
Instituto de Computação

EVOSUITE - Resultados

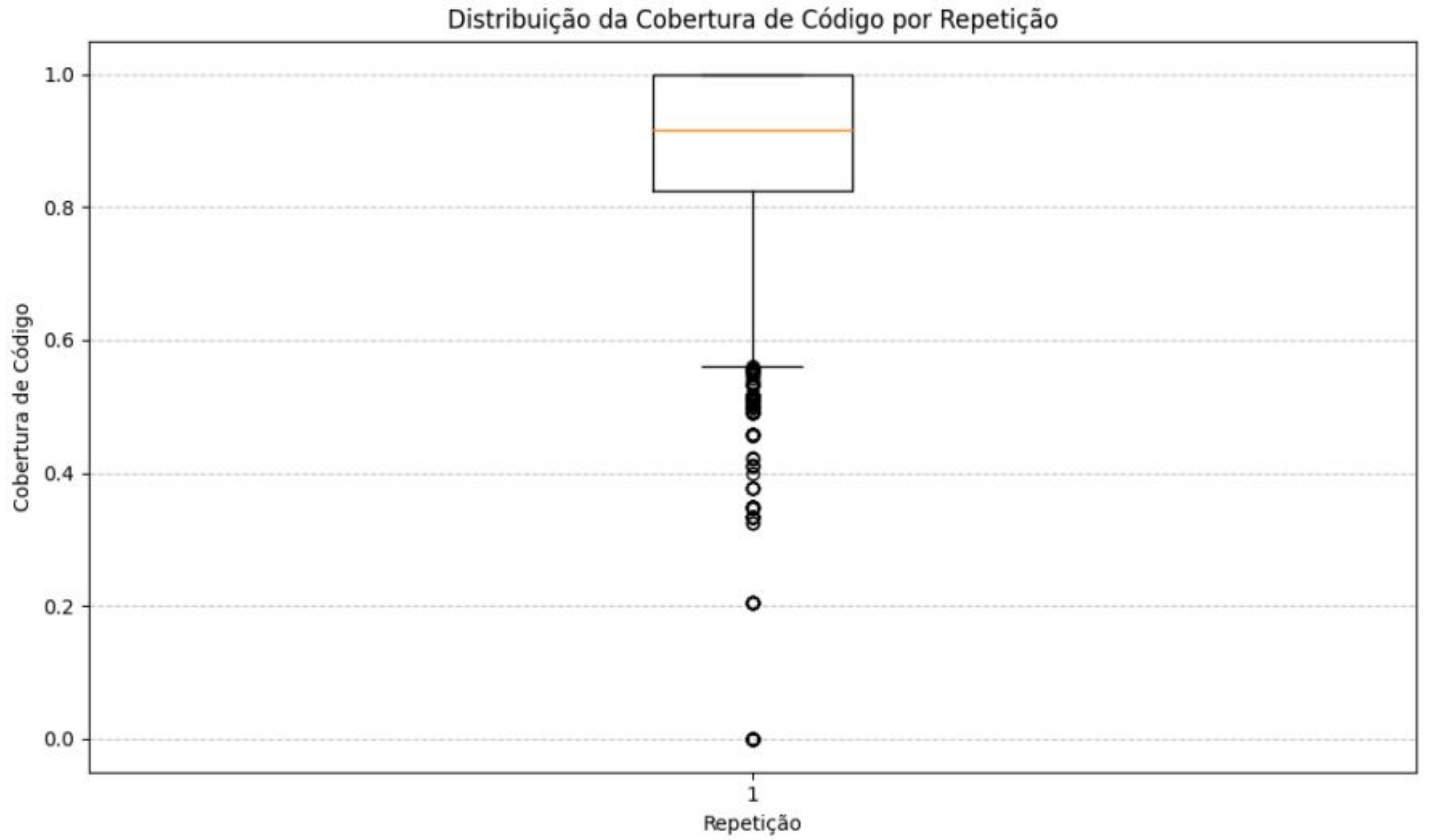


Figure 3. Cobertura por repetição

ICOMP
Instituto de Computação

EVOSUITE - Resultados

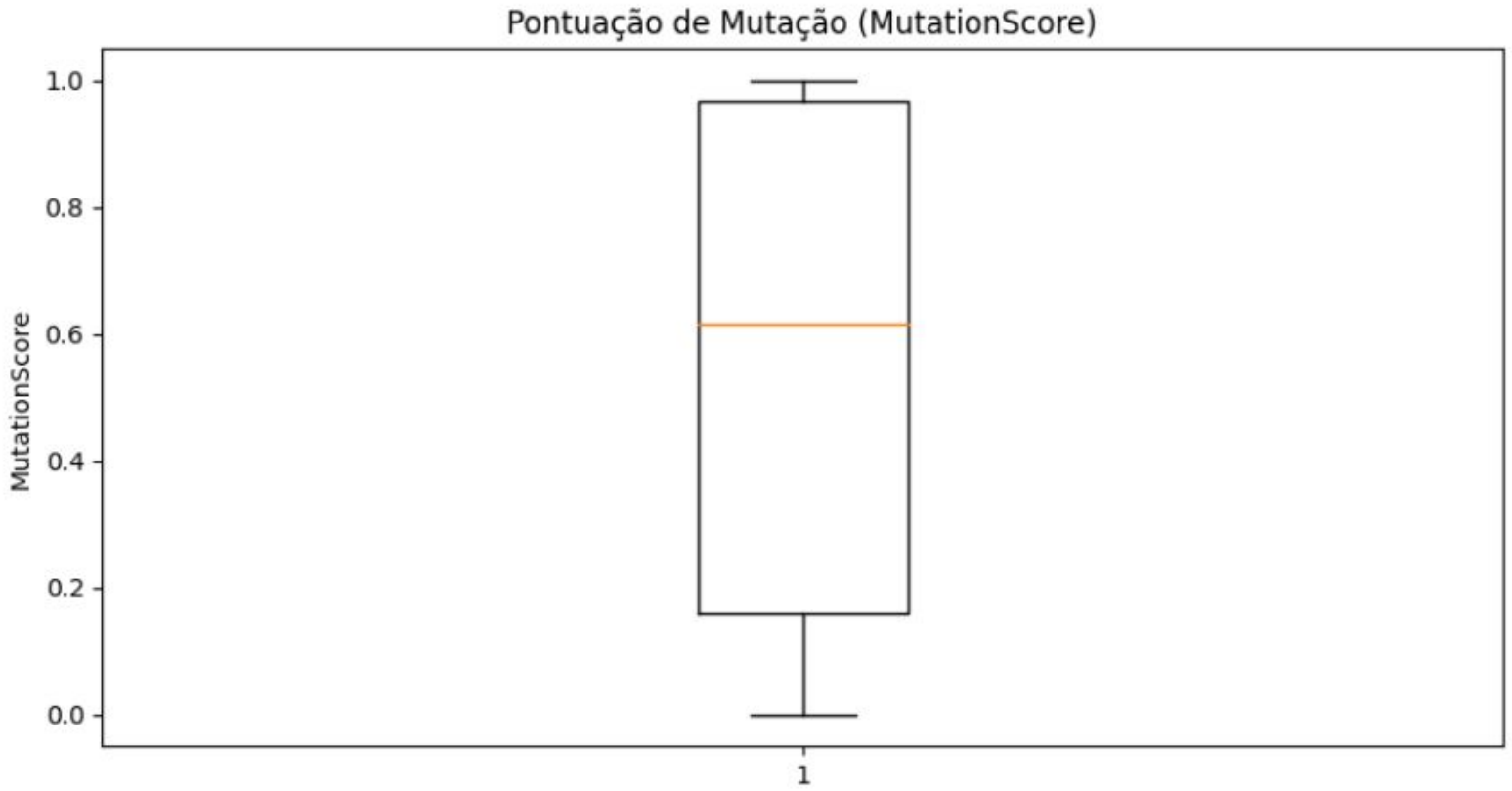


Figure 5. Pontuação de Mutação

ICOMP
Instituto de Computação

EVOSUITE - Resultados

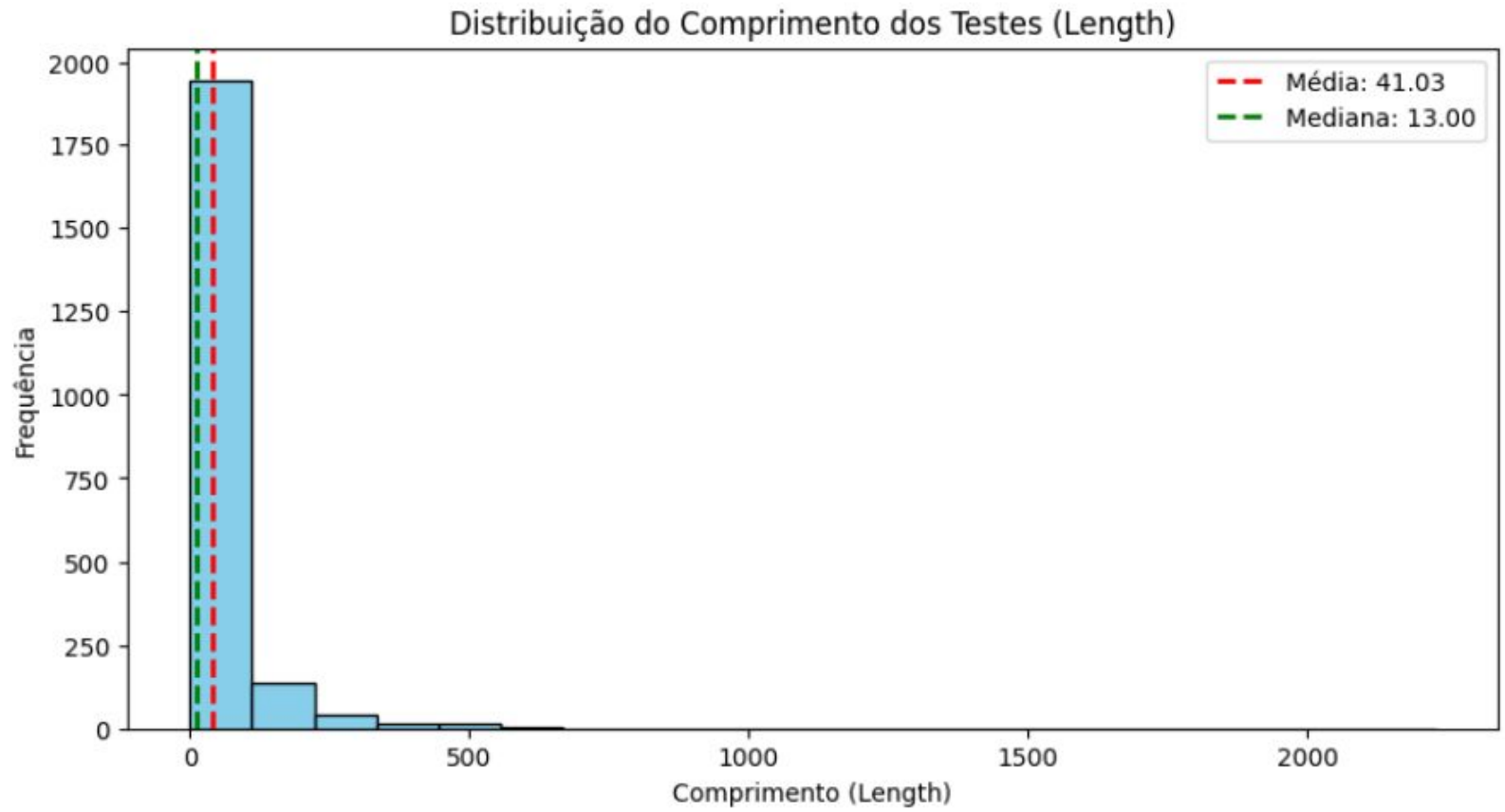


Figure 6. Comprimento dos Testes

ICOMP
Instituto de Computação

COLEMAN

Análise dos Resultados

ICOMP
Instituto de Computação

Datasets utilizados

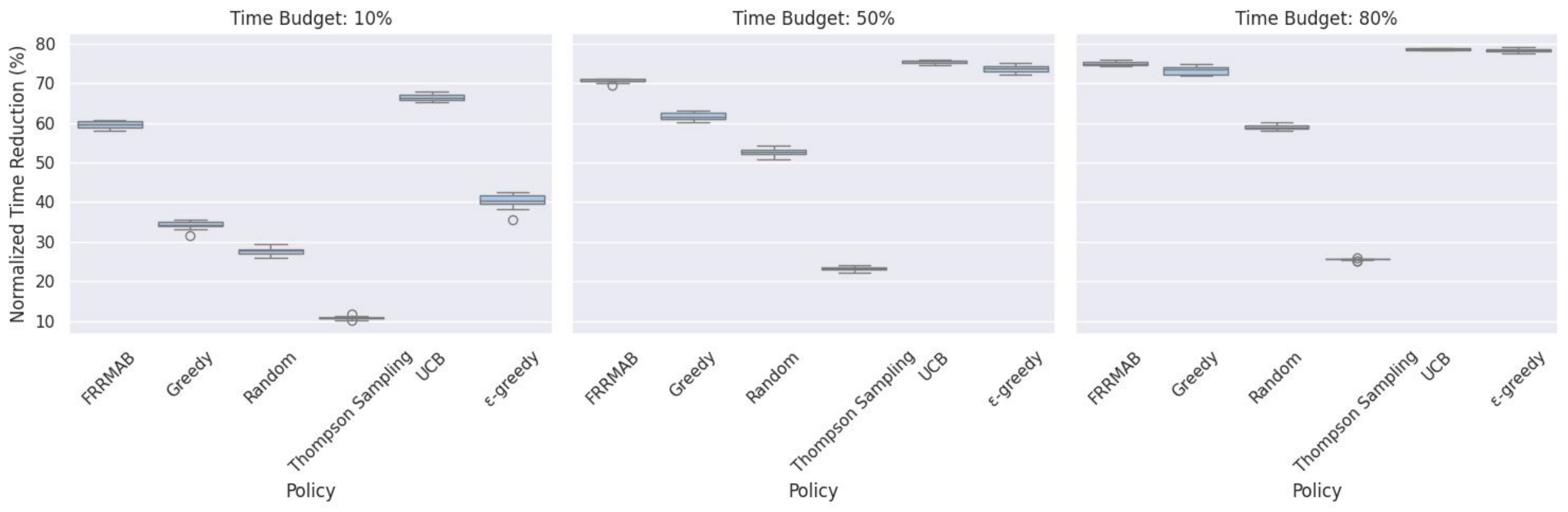
- **Druida**: Conjunto de dados para análise de comportamento de usuários em aplicativos móveis.
- **Fastjson**: Biblioteca de serialização que pode ser usada para converter objetos Java em JSON e vice-versa.
- **LexisNexis**: Empresa que fornece datasets jurídicos e de análise de risco para profissionais e organizações.
- **Deeplearning4j**: Framework de aprendizado profundo que oferece datasets para treinamento de modelos de inteligência artificial.

Distribuição Normalizada da Redução de Tempo

O gráfico da Figura 1 mostra a eficácia da estratégia Thompson Sampling na redução de tempo durante a priorização de testes em integração contínua. Embora consistente em orçamentos mais generosos, é importante considerar as restrições específicas de tempo ao escolher essa abordagem.

Distribuição Normalizada da Redução de Tempo

A política Thompson Sampling demonstra consistência em vários orçamentos de tempo (10%, 50% e 80%), adaptando-se bem às variações. No entanto, outras abordagens, como FRRMAB, são mais eficientes em orçamentos mais apertados. A Thompson Sampling é sólida e confiável, mas a escolha ideal depende dos trade-offs entre consistência e eficiência na tomada de decisões em integração contínua.

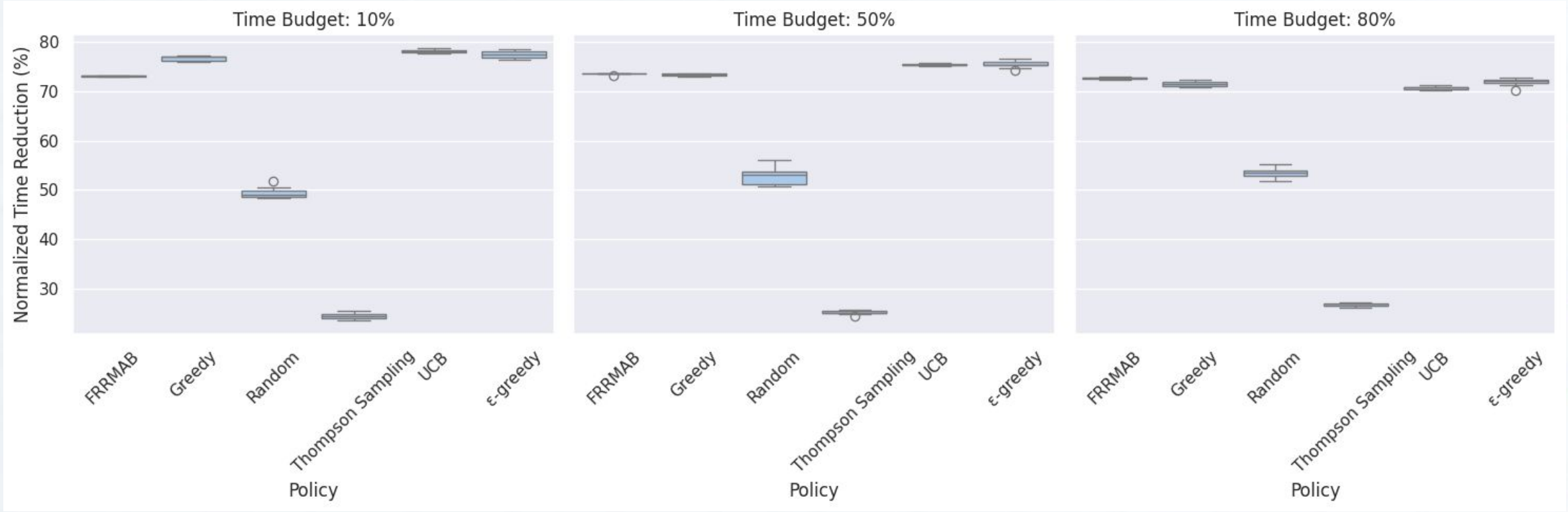


Distribuição Normalizada da Redução de Tempo

A Thompson Sampling revela consistência e eficiência na priorização de testes em integração contínua, com distribuições estáveis e alta redução de tempo normalizado. No entanto, sua menor exploratividade comparada à FRMAB pode ser desafiadora em espaços de tempo limitado.

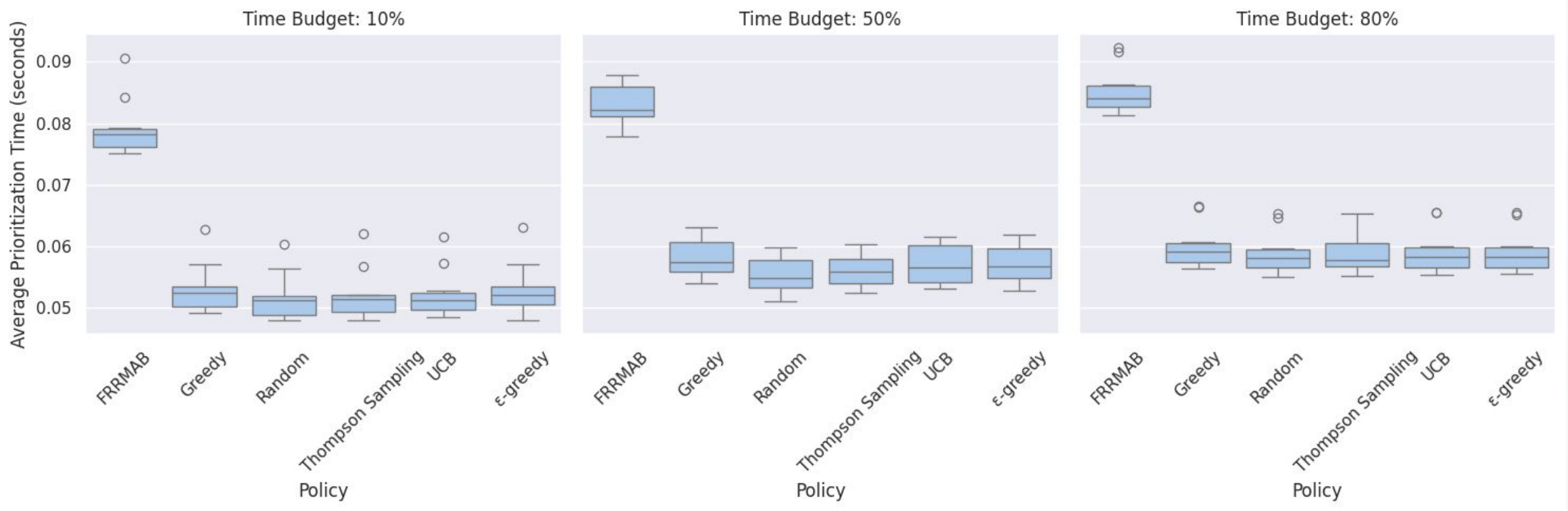
Distribuição Normalizada da Redução de Tempo

A Thompson Sampling exibe eficiência em orçamentos maiores (80%), com menor variabilidade nos resultados em comparação com outras estratégias. Entretanto, sua eficácia diminui em orçamentos menores, e em cenários críticos de tempo, pode não ser a melhor opção. A escolha da estratégia mais adequada para integração contínua deve considerar além dos dados apresentados, outros fatores do contexto.



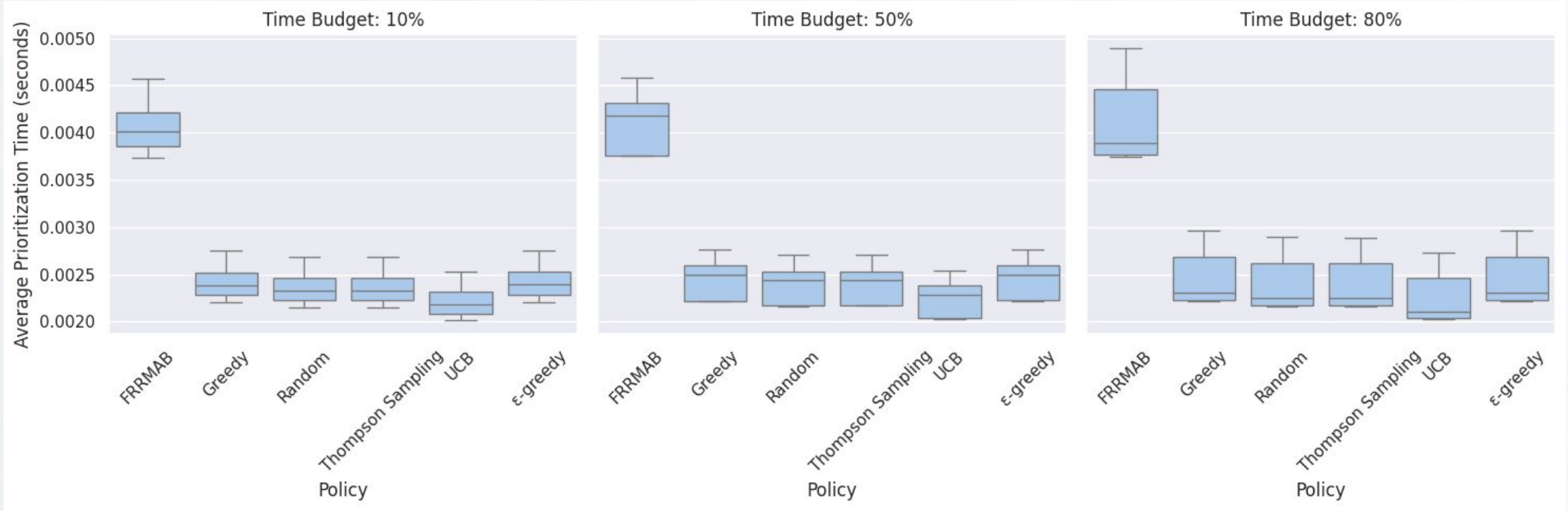
Priorização da Distribuição de Tempo

A política utilizada destaca-se pela eficiência de tempo, com menor tempo médio de priorização em todos os orçamentos de tempo. Sua consistência na performance é evidenciada pela baixa variação nos resultados. Porém, incertezas quanto à qualidade da priorização e outliers, especialmente no orçamento de 10%, requerem consideração. A decisão de adotar essa estratégia em integração contínua deve considerar não apenas a eficiência, mas também a qualidade das decisões tomadas.



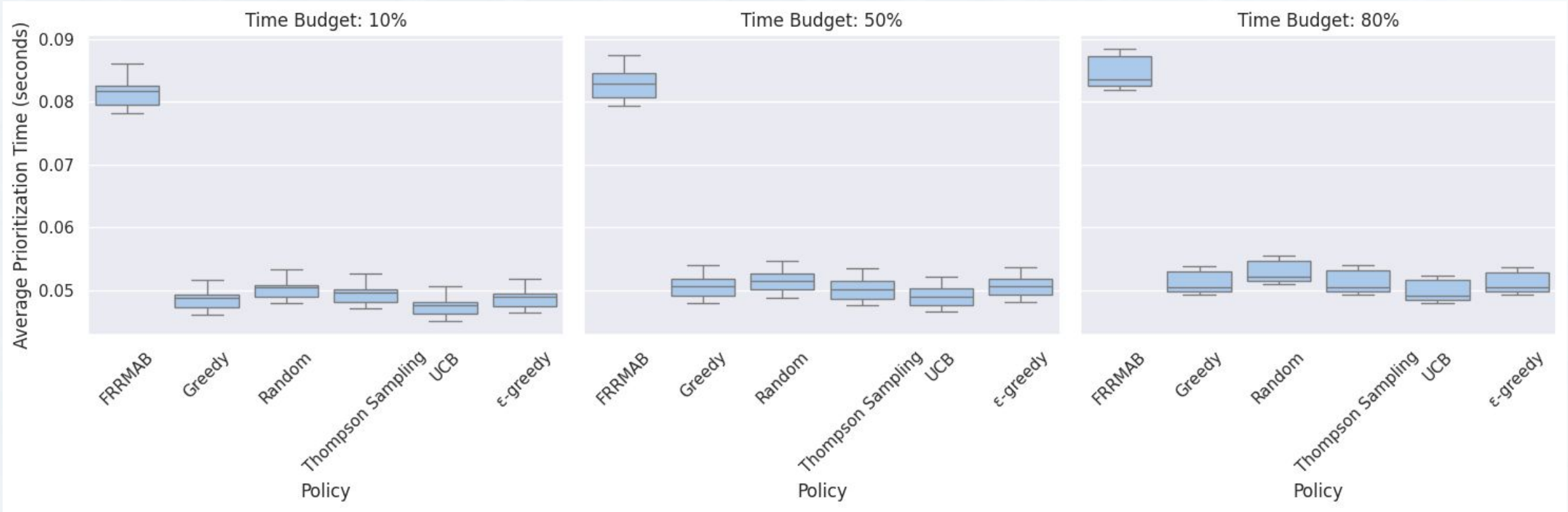
Priorização da Distribuição de Tempo

Na Figura os resultados do Thompson Sampling destacam sua eficiência consistente, mantendo baixo tempo de priorização em diferentes orçamentos de tempo, indicando robustez em diversos cenários. No entanto, sua performance semelhante à estratégia UCB em orçamentos maiores e ausência de destaque drástico em relação às outras políticas sugerem vantagens mais notáveis em situações específicas. O gráfico compara o "Average Prioritization Time" entre cinco políticas diferentes sob três "Time Budgets", oferecendo insights sobre o desempenho relativo das estratégias em diferentes contextos de orçamento de tempo.



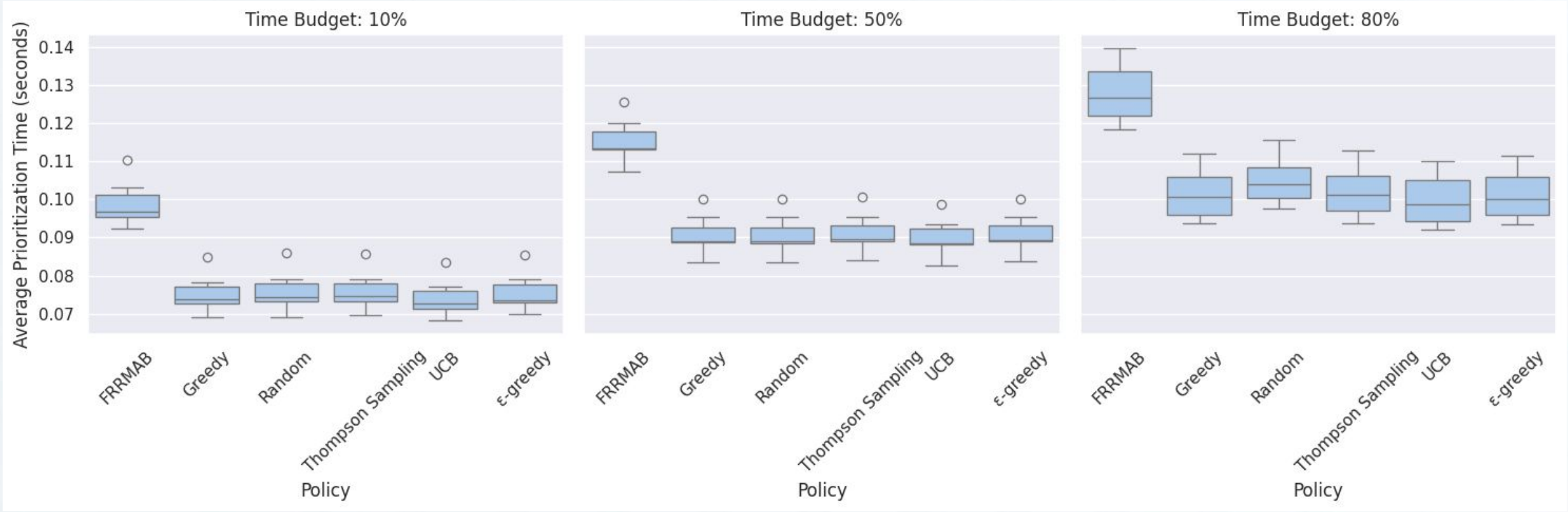
Priorização da Distribuição de Tempo

Nesta Figura, a política Thompson Sampling destaca-se pela eficiência, com baixo tempo médio de priorização em todos os orçamentos de tempo, e consistência, mantendo tempos baixos independentemente do orçamento. Porém, não é a mais rápida em todos os cenários, sendo superada pelo FRMAB em um orçamento de 80%, e apresenta maior variação nos resultados, especialmente em orçamentos mais limitados, sugerindo possíveis inconsistências na performance. As outras estratégias (FRMAB, Greedy, Random, UCB e ε-Greedy) exibem tempos médios de priorização variados, com o FRMAB mostrando a menor média no orçamento de 80%.



Priorização da Distribuição de Tempo

Com base na Figura abaixo a política Thompson Sampling revela consistência e equilíbrio entre eficiência e estabilidade na priorização, embora não seja a mais rápida em todos os cenários. Em orçamentos mais baixos, como 10%, outras estratégias como Greedy ou FRAMAB podem se destacar mais. No entanto, considerando o contexto e objetivos específicos, a Thompson Sampling oferece uma escolha sólida para equilibrar eficiência e consistência na priorização.

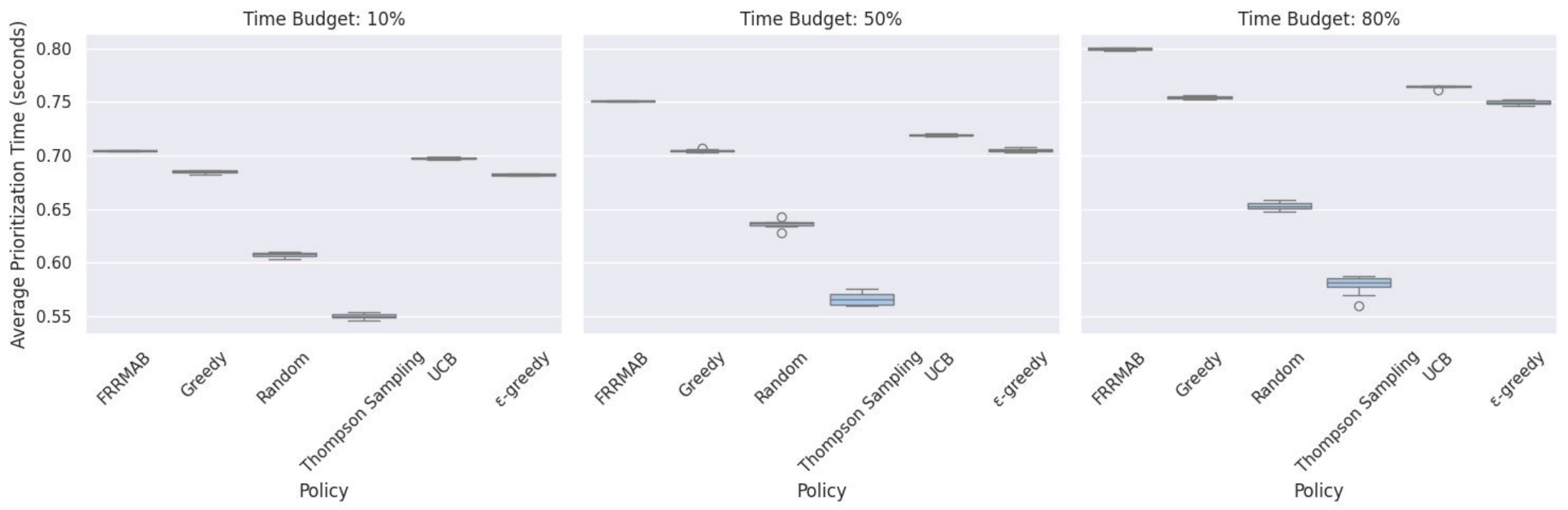


Distribuição NAPFD - avg_fitness_time

É possível destacar nos gráficos quanto a política Thompson Sampling sua eficiência em espaços de tempo menores e consistência em espaços de tempo médios, embora possa ser superada por outras estratégias em orçamentos maiores. A escolha da estratégia ideal depende das prioridades específicas e dos trade-offs entre eficiência e exploração.

Distribuição NAPFD

A Thompson Sampling, nos gráficos, destaca-se por seu desempenho superior ao Random e comparável ao Greedy em orçamentos menores, mas é superada pelo FRAMAB e Greedy em orçamentos maiores. As outras estratégias também exibem seus próprios perfis de desempenho. Em resumo, a Thompson Sampling mostra-se promissora em cenários de médio prazo, mas sua eficácia pode variar dependendo do contexto e do orçamento de tempo disponível.

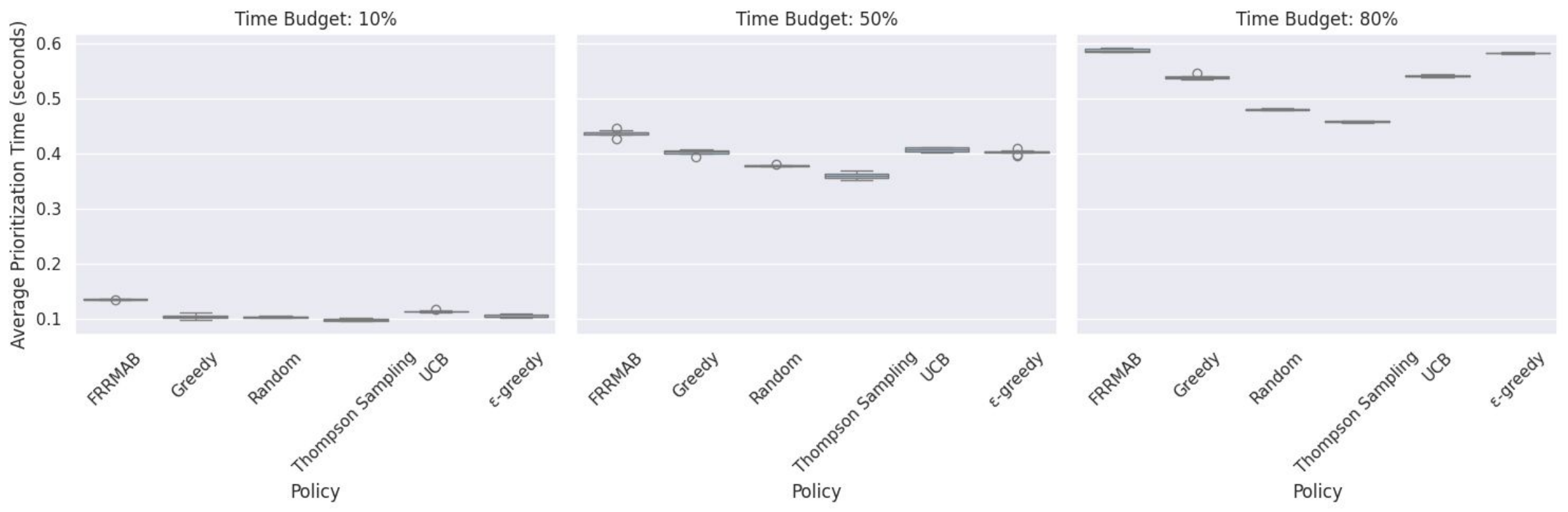


Distribuição NAPFD

Na FIgura a Thompson Sampling destaca-se por seu desempenho consistente em diferentes orçamentos de tempo, com tempo de priorização mais baixo que o FRAMAB e o Random em todos os cenários. No entanto, é superada pelo Greedy e ε-greedy em todos os cenários e não mostra melhoria significativa com o aumento do orçamento de tempo. Greedy e ε-greedy emergem como opções competitivas para priorização de testes em integração contínua.

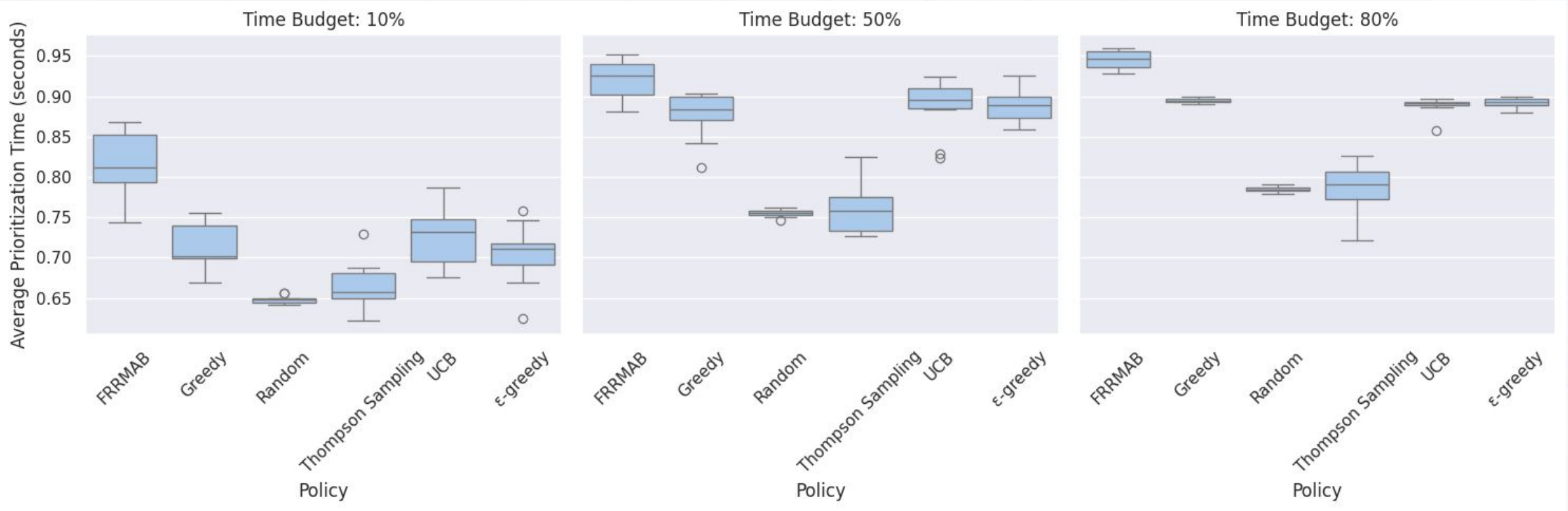
Distribuição NAPFD

Nos gráficos, a Thompson Sampling destaca-se pela menor variação nos resultados, indicando maior estabilidade. No entanto, não é a estratégia mais rápida, sendo superada por outras como Random e UCB em orçamentos maiores. Apesar disso, oferece consistência e estabilidade na integração contínua, destacando a importância de equilibrar tempo e precisão na seleção da estratégia ideal.



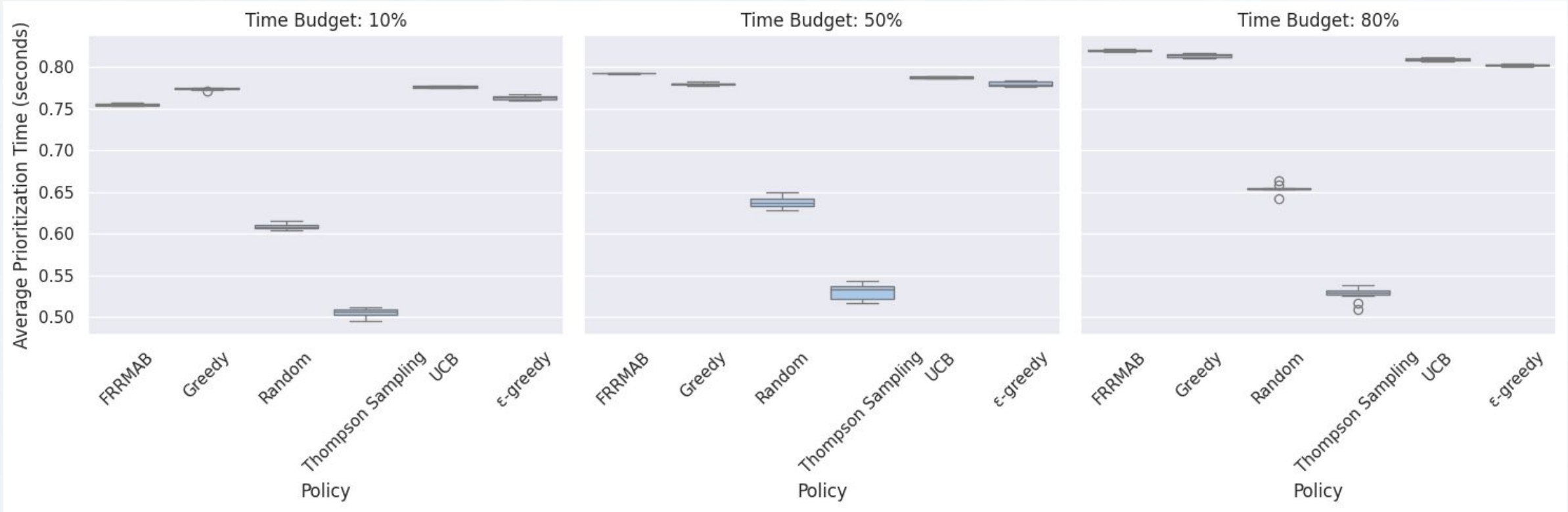
Distribuição APFDc - avg_cost_time

A política Thompson Sampling destaca-se por sua superioridade sobre as estratégias Greedy e Random e um desempenho significativamente melhor que todas as outras estratégias em orçamentos médios. No entanto, seu desempenho cai no orçamento de tempo de 80%, e a variabilidade nos resultados sugere que pode não ser a estratégia mais confiável em todos os contextos. Enquanto isso, o FRAMAB é consistente em todos os orçamentos, o Greedy tem um desempenho mediano, e o Random mostra resultados aleatórios.



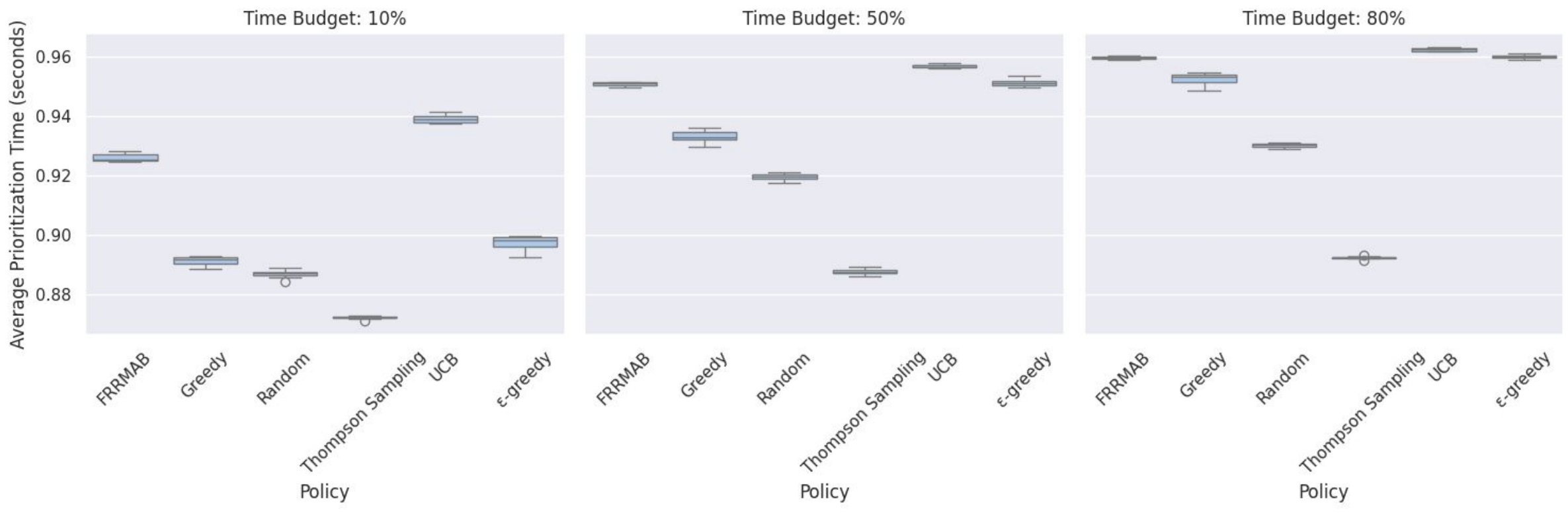
Distribuição APFDc

A política Thompson Sampling, na Figura, destaca-se com desempenho comparável ao Random e superior ao Greedy e ε-Greedy em orçamentos de tempo menores. Em orçamentos de tempo maiores, supera todas as outras estratégias, porém seu desempenho cai drasticamente em orçamentos de tempo maiores. A inconsistência nos resultados pode limitar sua aplicação em diferentes contextos, ressaltando a importância de considerar as características de cada estratégia ao priorizar testes em integração contínua.



Distribuição APFDc

A Thompson Sampling não se destaca como a estratégia mais eficaz em espaços de tempo de 10% e 50%, ficando atrás do FRAMAB e do Greedy. No entanto, no orçamento de tempo de 80%, supera todas as outras estratégias em termos de APFDc. A variação nos resultados destaca a necessidade de considerar cuidadosamente os prós e contras ao aplicar a Thompson Sampling em um contexto específico.



Distribuição APFDc

A Thompson Sampling mantém uma priorização estável em diferentes orçamentos de tempo, superando Greedy e Random em todos os cenários. Sua capacidade de exploração adaptativa permite rápida correção e adaptação às mudanças no ambiente. No entanto, seu custo computacional e desempenho relativo em orçamentos maiores devem ser considerados ao decidir sua aplicação.

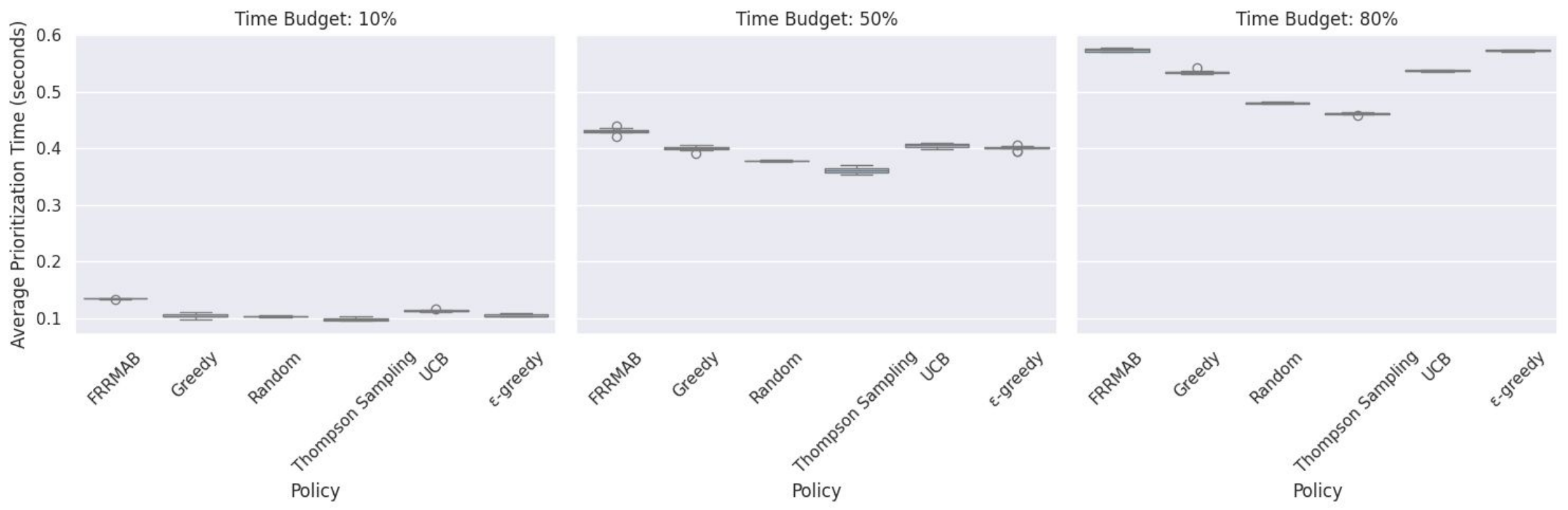


Tabela - Priorização de Tempo

Na Tabela abaixo, os tempos de priorização médios da Thompson Sampling são comparáveis aos das outras estratégias em todos os três espaços de tempo, sugerindo uma exploração mais ampla. No entanto, a Thompson Sampling leva mais tempo para priorizar os testes em comparação com outras estratégias, o que pode ser uma desvantagem em cenários com restrições de tempo. Em resumo, suas vantagens em exploração e eficácia em cenários complexos podem torná-la uma escolha valiosa em contextos específicos.

TimeBudget	Random	UCB	Greedy	ε-greedy	FRRMAB	Thompson Sampling
0.1	0.0460	0.0452	0.0461	0.0461	0.0681	0.0461
0.5	0.0507	0.0507	0.0516	0.0512	0.0732	0.0508
0.8	0.0552	0.0536	0.0547	0.0542	0.0769	0.0543

Tabela - NAPFD

Na Tabela abaixo, a política Thompson Sampling apresenta tempos médios de aptidão comparáveis em todos os três orçamentos de tempo analisados. Destaca-se por equilibrar efetivamente exploração e explotação de ações, adaptando suas crenças de forma adaptativa. Embora ofereça benefícios importantes, a escolha da estratégia mais adequada deve considerar cuidadosamente o cenário em questão.

TimeBudget	Random	UCB	Greedy	ε-greedy	FRRMAB	Thompson Sampling	Metric
0.1	0.5789	0.6281	0.6109	0.6135	0.6574	0.5642	avg_fitness_time
0.5	0.6859	0.7542	0.7414	0.7564	0.7832	0.6662	avg_fitness_time
0.8	0.7251	0.7980	0.7972	0.8104	0.8370	0.6932	avg_fitness_time

Tabela - APFDc

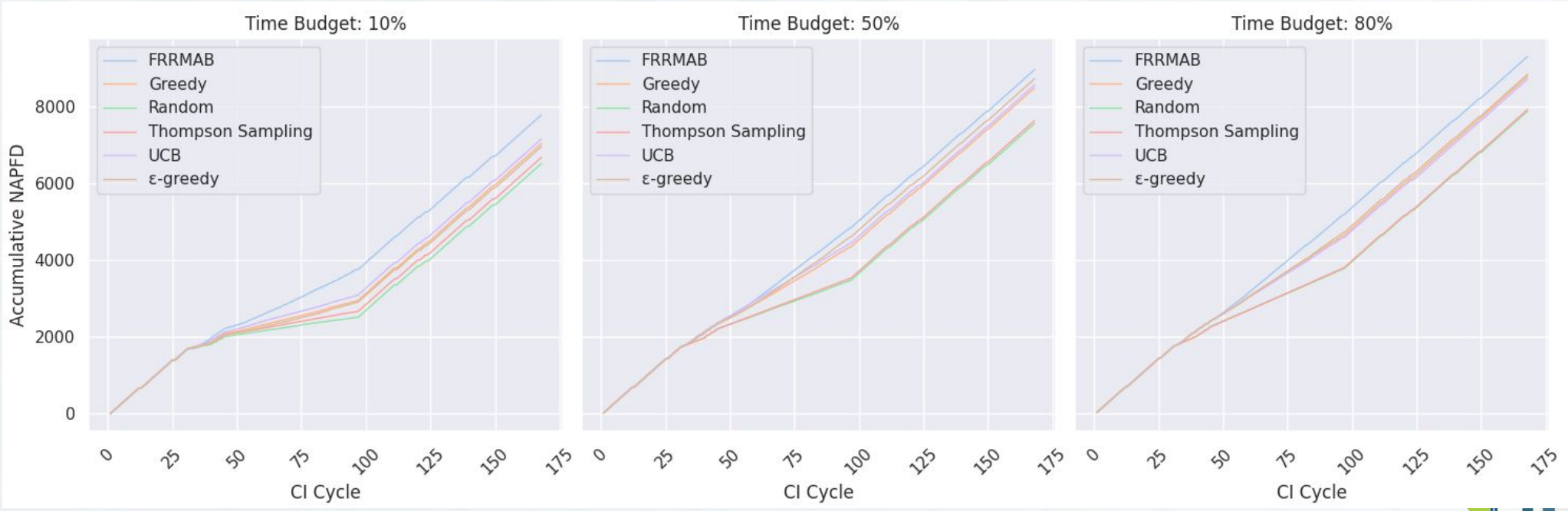
Na Tabela podemos observar que a Thompson Sampling demonstra um desempenho superior em termos de tempo médio de custo, especialmente em cenários com maiores valores de TimeBudget (0.5 e 0.8). Sua abordagem adaptativa, considerando a incerteza sobre as recompensas das ações, é vantajosa em ambientes dinâmicos. No entanto, sua implementação pode ser mais intensiva computacionalmente e sua eficácia depende da qualidade das estimativas das distribuições de recompensa. Assim, a Thompson Sampling requer uma consideração cuidadosa dos trade-offs e da complexidade computacional ao ser implementada.

TimeBudget	Random	UCB	Greedy	ε-greedy	FRRMAB	Thompson Sampling	Métrica
0.1	0.5794	0.6488	0.6354	0.6355	0.6821	0.5556	**avg_cost_time**
0.5	0.6881	0.7824	0.7703	0.7805	0.8021	0.6611	**avg_cost_time**
0.8	0.7265	0.8158	0.8179	0.8251	0.8467	0.6834	**avg_cost_time**

ICOMP
Instituto de Computação

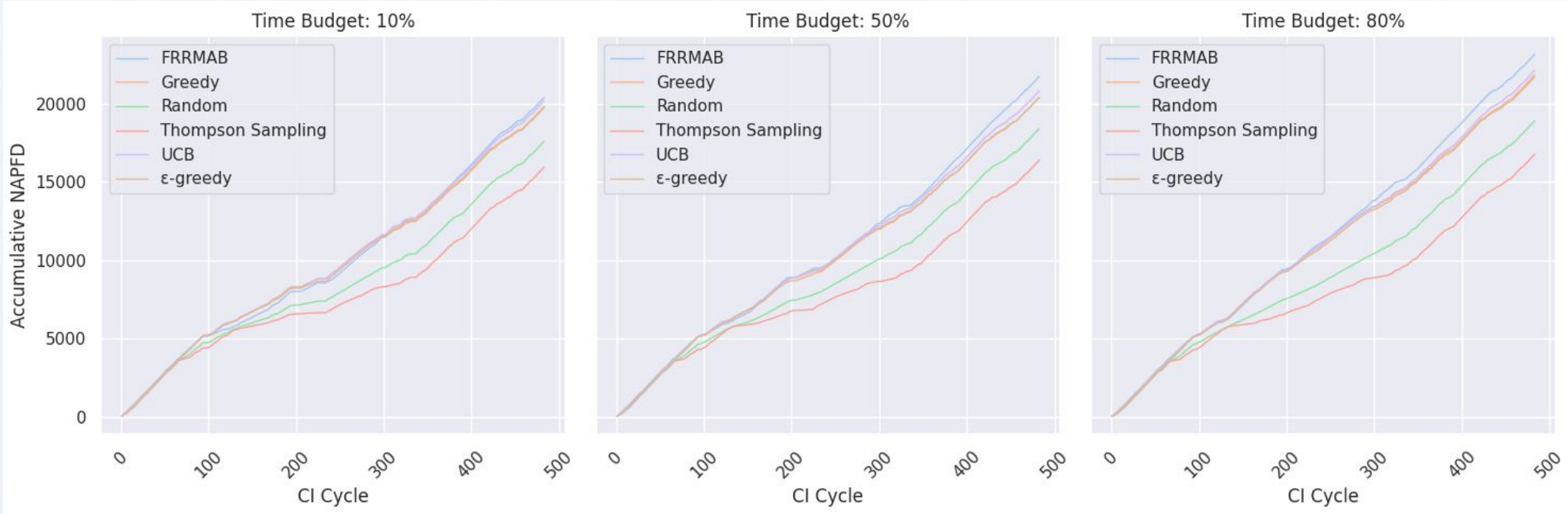
NAPFD com algoritmo cum fitness

Na figura abaixo o algoritmo fitness variation com o conjunto de dados druid. Destaque para o Thompson Sampling, com desempenho notável. O método FRRMAB supera consistentemente outras estratégias, especialmente em cenários com mais recursos disponíveis.



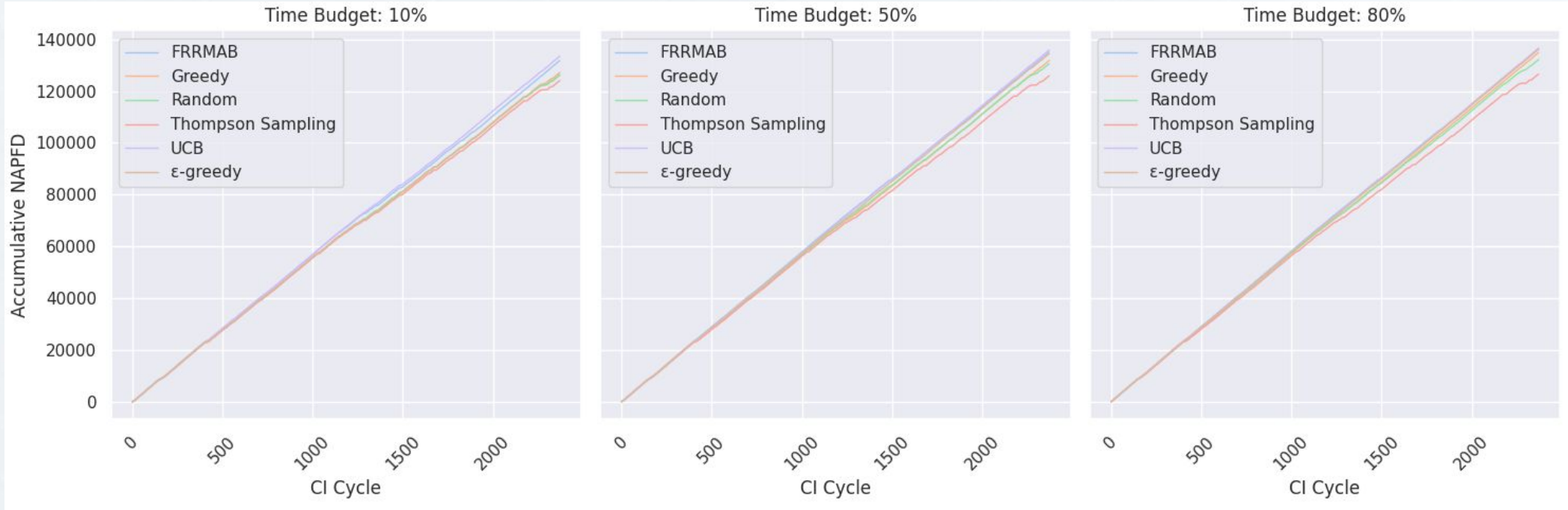
NAPFD com algoritmo cum fitness

Na figura abaixo a análise de dados com o algoritmo fitness variation e o conjunto de dados 'deeplearning4j'. O FRRMAB demonstrou consistentemente o melhor desempenho, seguido pelo Greedy. Embora UCB e ε-greedy apresentem padrões semelhantes, o ε-greedy mostra um desempenho ligeiramente superior, ambos superando a abordagem aleatória, mas abaixo de outras estratégias na identificação da melhor ação.



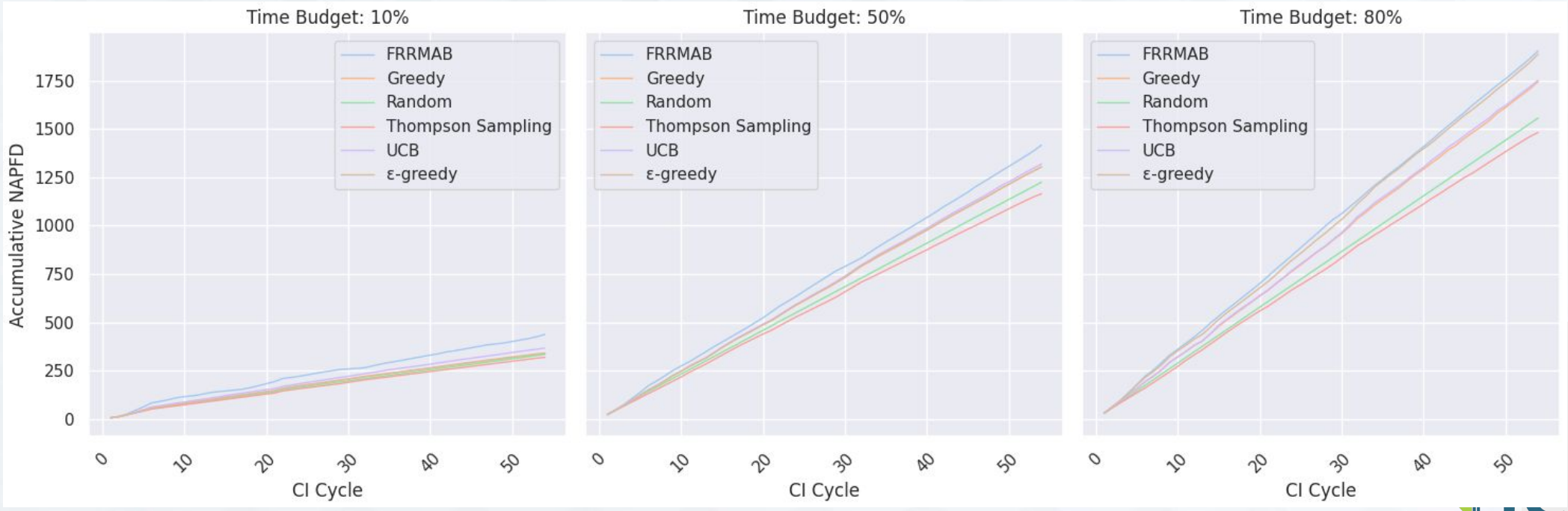
NAPFD com algoritmo cum fitness

Na figura abaixo a análise dos dados com o algoritmo fitness variation e o conjunto de dados 'fatjson' revelou tendências significativas no desempenho das estratégias de seleção. Para 10% do Orçamento de Tempo, FRRMAB se destacou como a mais eficaz, seguida por Greedy, UCB, Thompson Sampling, ε-greedy e Random. À medida que o orçamento de tempo aumenta para 80%, a ordem de desempenho permanece relativamente constante em comparação com o segundo gráfico analisado.



NAPFD com algoritmo cum fitness

Na figura abaixo a análise de dados com algoritmo fitness variation e conjunto 'lexisNexis'. Destaque para três estratégias de seleção de braço em bandits multi-armed. Greedy tende à sub-otimização, UCB equilibra exploração e exploração, superando o Thompson Sampling em certos cenários. ε-greedy oferece abordagem simplificada, porém menos eficaz que Thompson Sampling. Essas análises destacam a importância de escolher estratégias adequadas em problemas de bandit multi-armed.



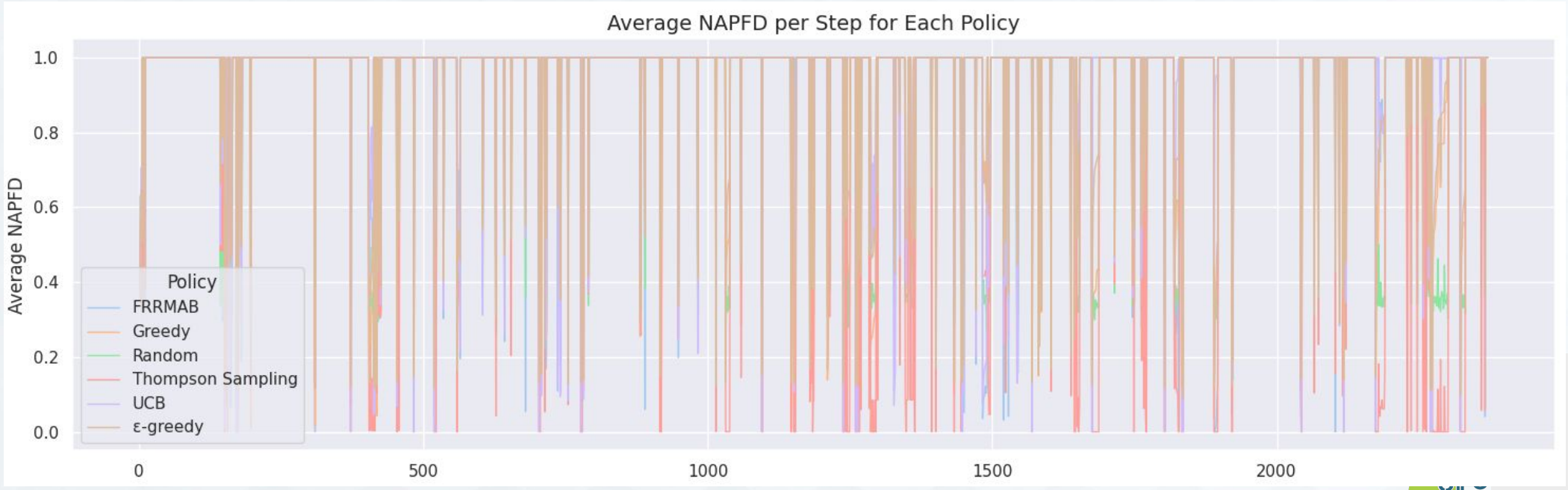
Instituto de Computação

NAPFD com algoritmo fitness variation

Análise de dados utilizando o algoritmo fitness variation com o conjunto de dados druid.

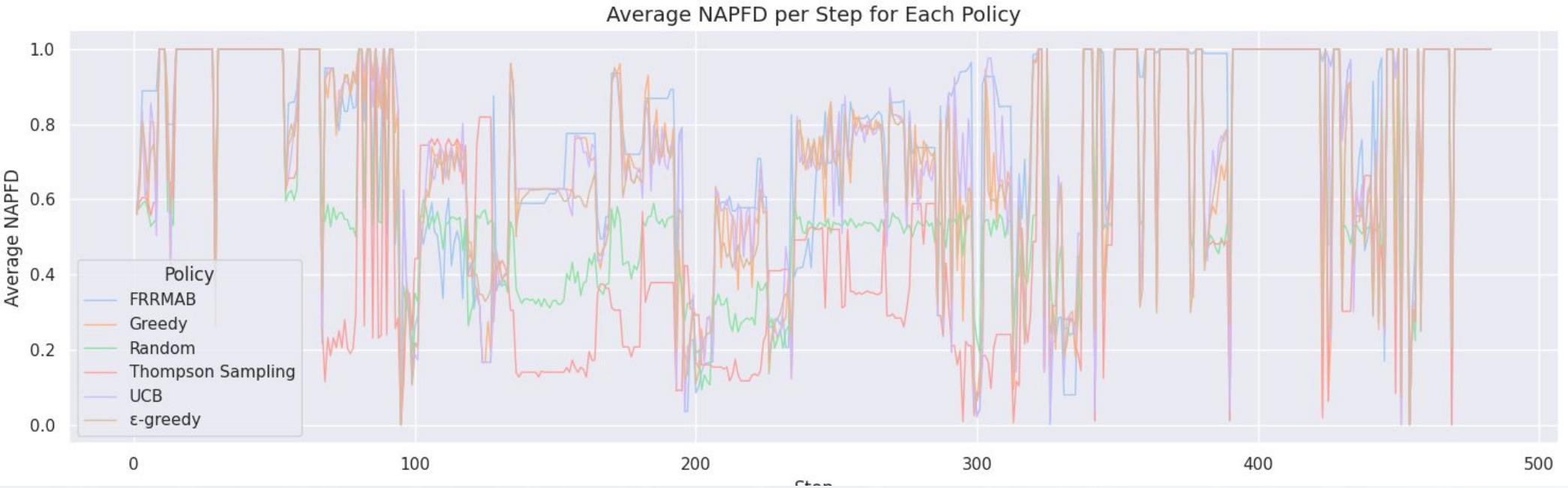
NAPFD com algoritmo fitness variation

Análise de dados utilizando o algoritmo fitness variation com o conjunto de dados fatjson.



NAPFD com algoritmo fitness variation

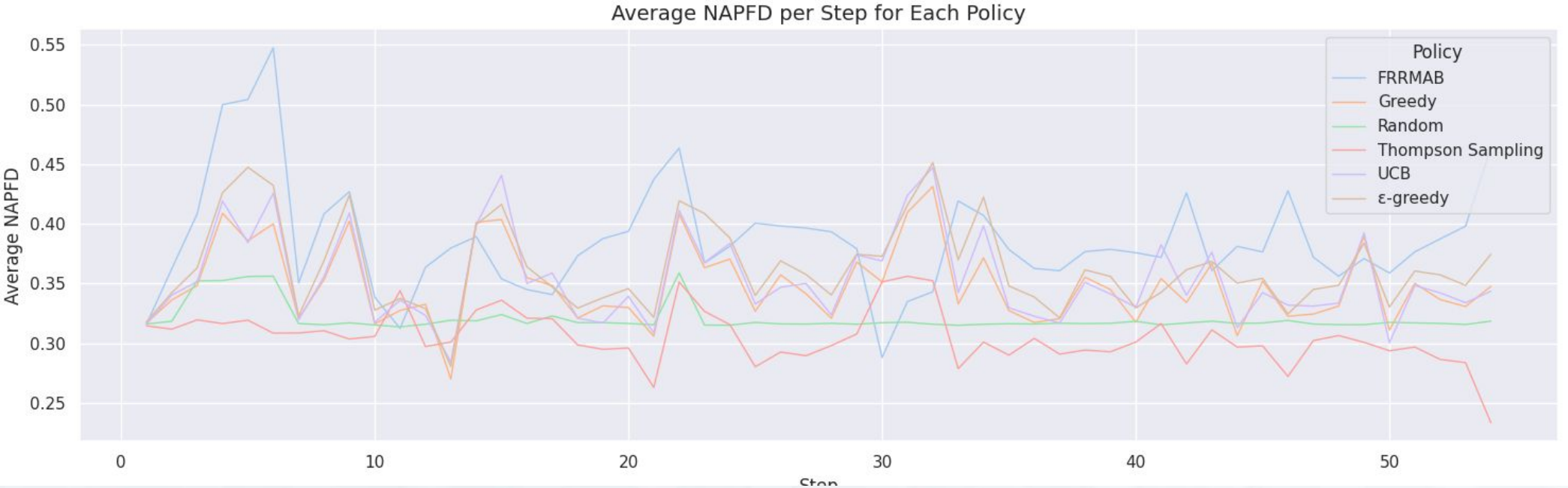
Análise de dados utilizando o algoritmo fitness variation com o conjunto de dados deeplearning4j.



Instituto de Computação

NAPFD com algoritmo fitness variation

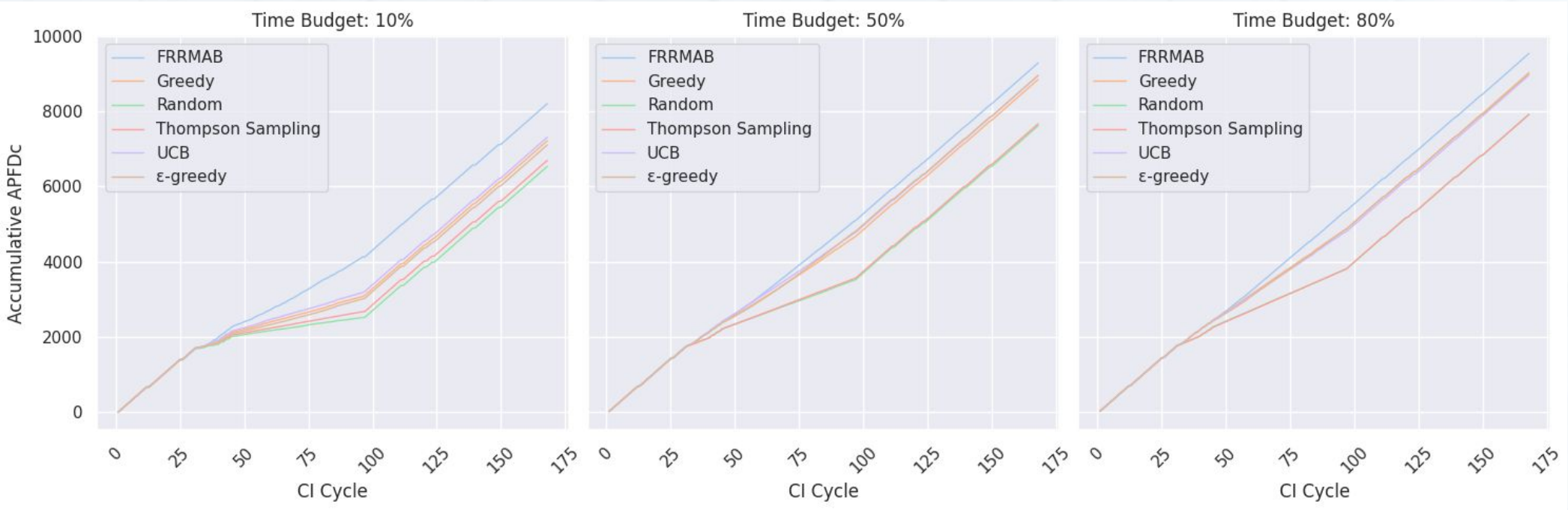
Análise de dados utilizando o algoritmo fitness variation com o conjunto de dados lexisNexis



ICOMP
Instituto de Computação

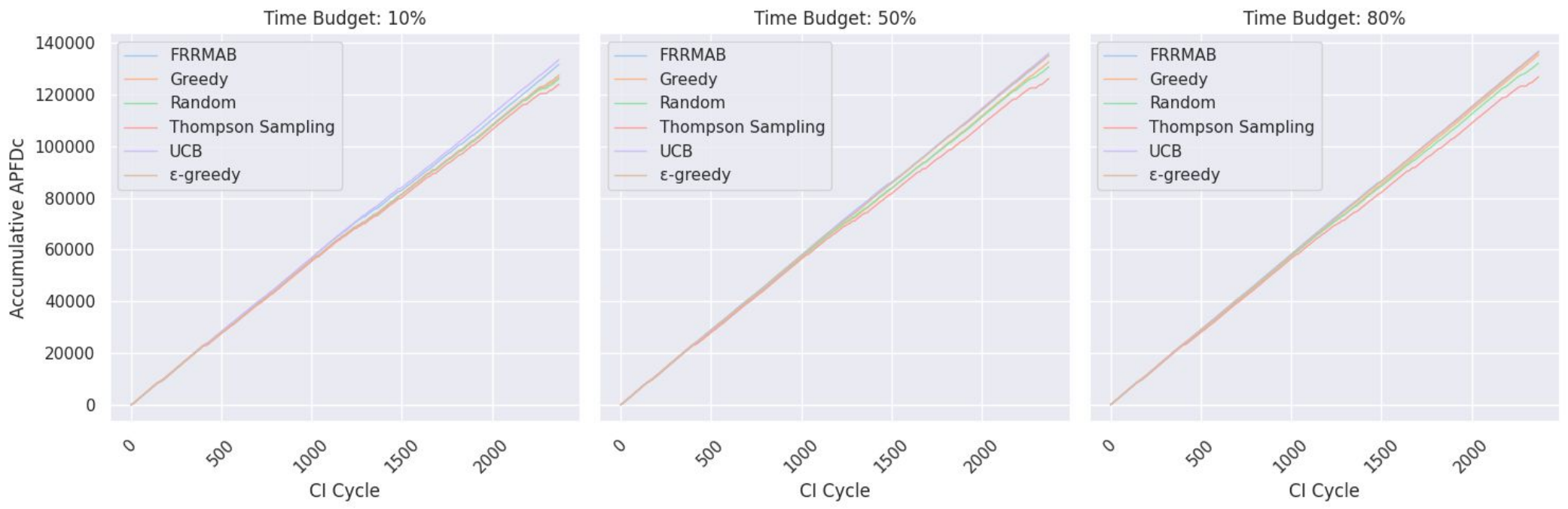
APFDc com algoritmo cum cost

Análise de dados utilizando o algoritmo cum cost com o conjunto de dados druid



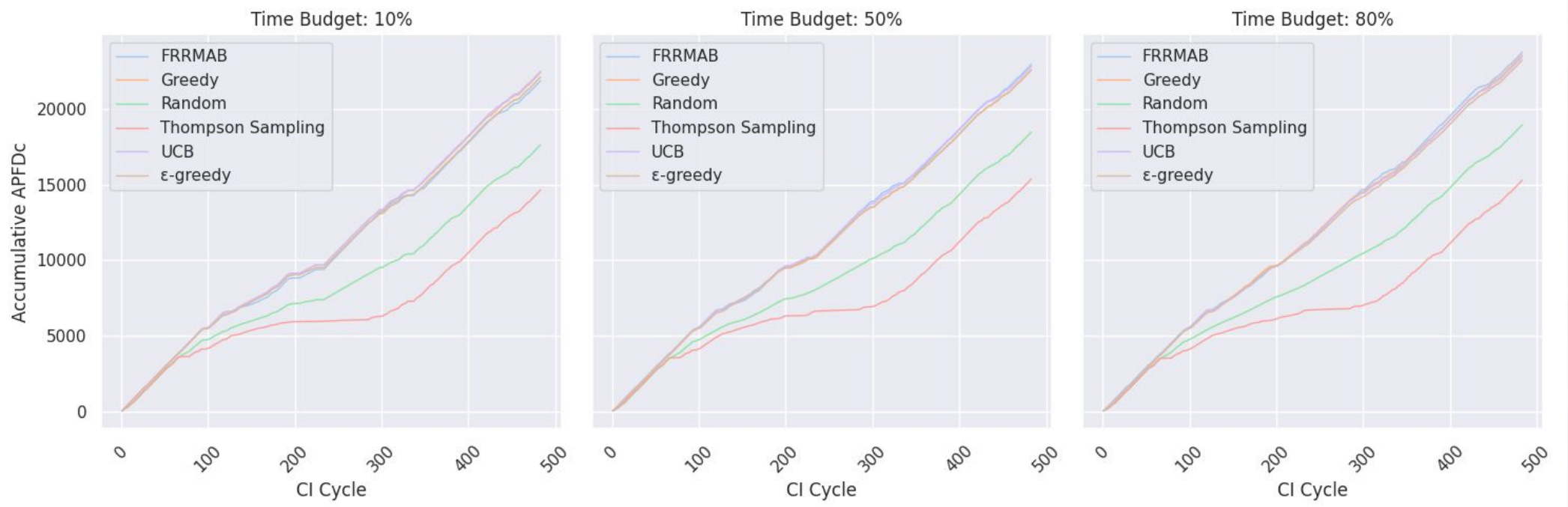
APFDc com algoritmo cum cost

Análise de dados utilizando o algoritmo cum cost com o conjunto de dados fatjson



APFDc com algoritmo cum cost

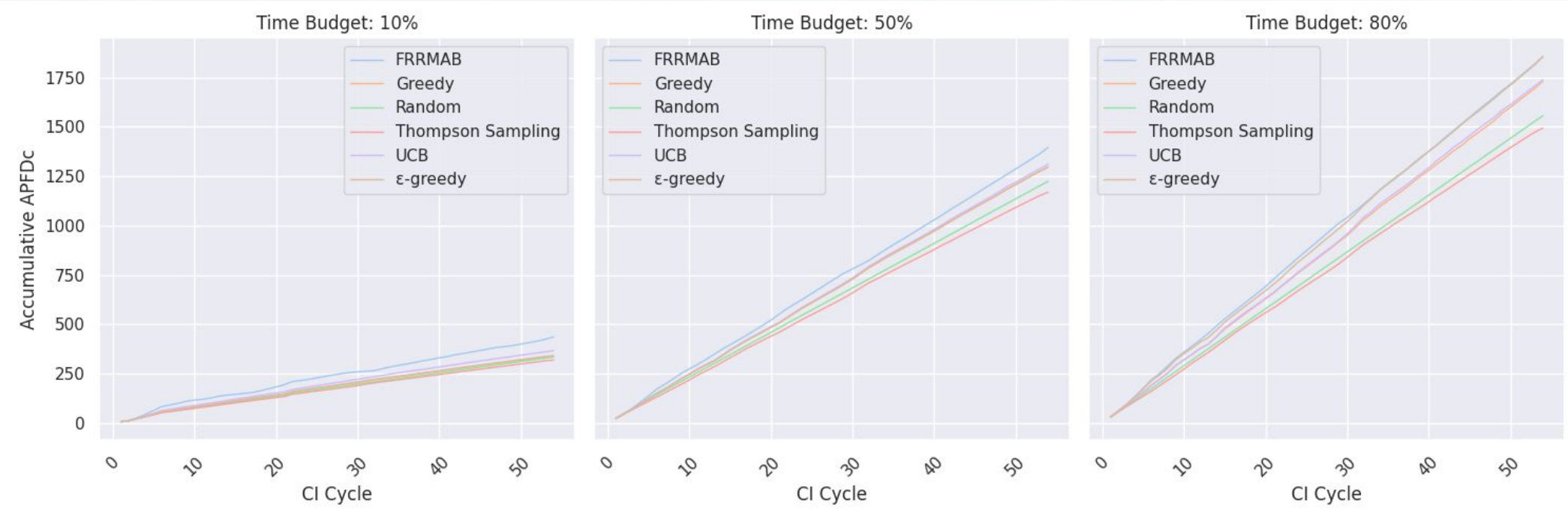
Análise de dados utilizando o algoritmo cum cost com o conjunto de dados deeplearning4j



Instituto de Computação

APFDc com algoritmo cum cost

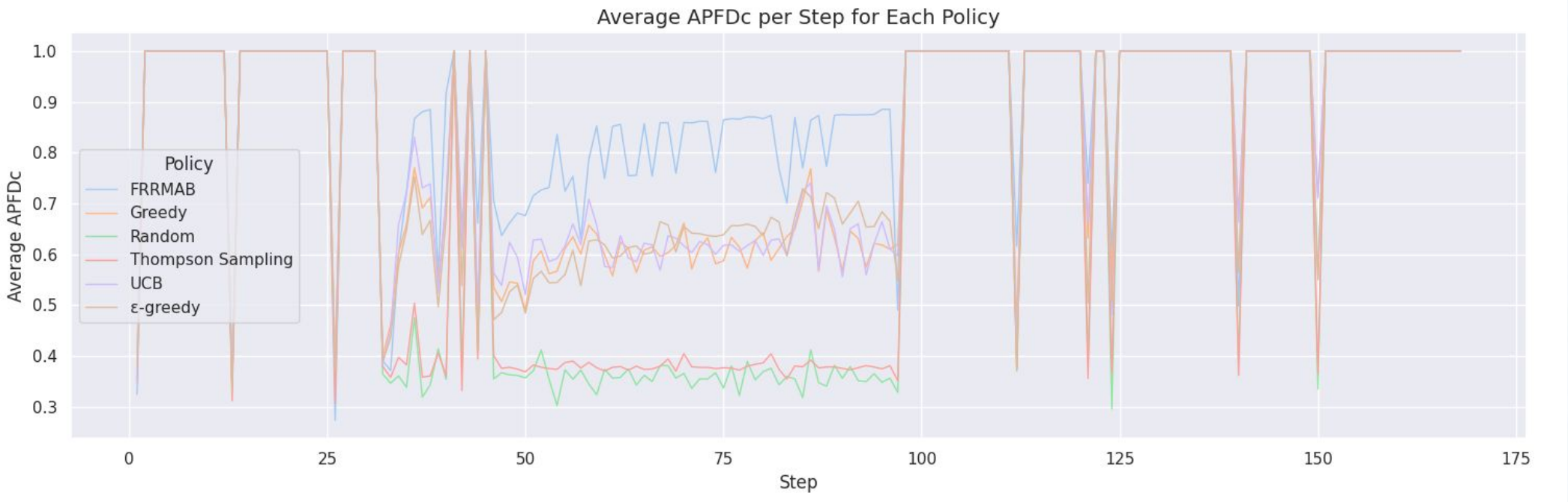
Análise de dados utilizando o algoritmo cum cost com o conjunto de dados lexisNexis



Instituto de Computação

APFDc com algoritmo cost variation

Análise de dados utilizando o algoritmo cost variation com o conjunto de dados druid



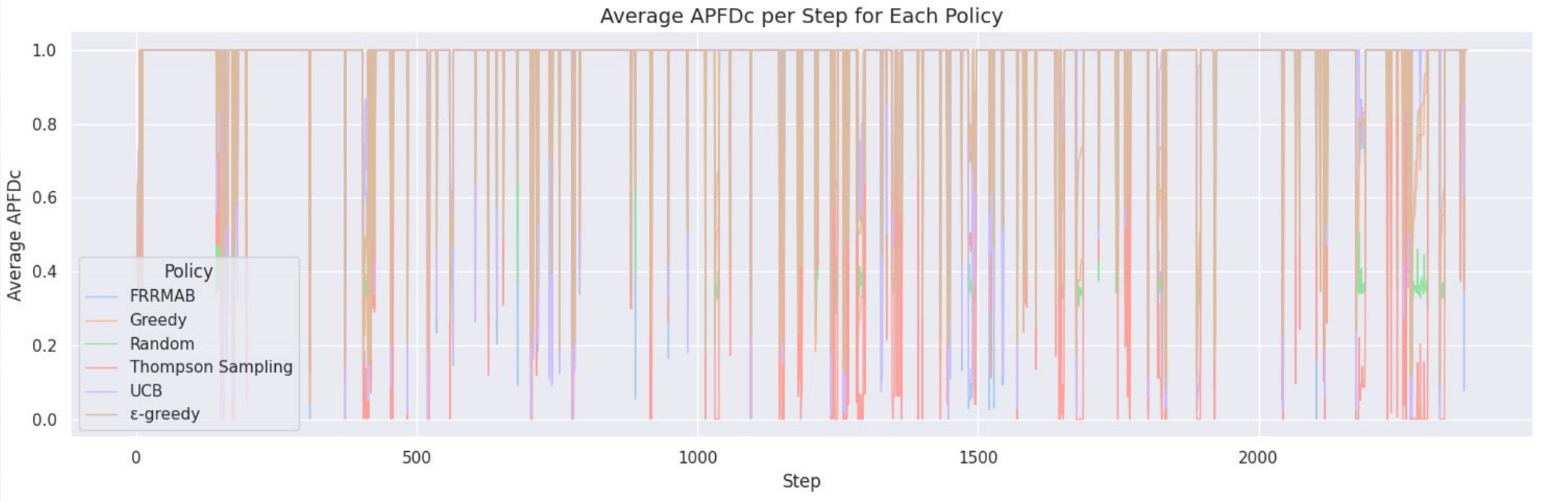
ICOMP
Instituto de Computação

APFDc com algoritmo cost variation

Análise de dados utilizando o algoritmo cost variation com o conjunto de dados deeplearning4j

APFDc com algoritmo cost variation

Análise de dados utilizando o algoritmo cost variation com o conjunto de dados fatjson



APFDc com algoritmo cost variation

Análise de dados utilizando o algoritmo cost variation com o conjunto de dados lexisNexi

ICOMP
Instituto de Computação