دانشگاه صنعتی شریف

دانشکده مهندسی کامپیوتر

پیاده سازی **Cache Replacement Imitating Belady's**

OPT Policy در **Gem5**

استاد: دکتر اسدی

نیمسال تحصیلی: بهار ۱۴۰۳

اعضای تیم

آریان افضل زاده

ملیکا علیزاده

الینا هژبری

فاطمه حمدی

چکیده مقاله

در این پروژه می خواهیم سیاست Belady's MIN را پیاده سازی کنیم. طبق مقاله این سیاست باید به آخرین سطح cache اعمال شود تا خطی که پیش بینی شده در آینده دوباره استفاده می شود را خارج کند. زمان تقریبی رسیدن به هر خط (ETA) برابر جمع زمان کنونی و پارامتری به اسم Predicted Reuse Distance است. در هر درج خط با بزرگترین ETA خارج می شود. همچنین برای پیش بینی reuse distance از یک پیش بینی کننده بر اساس PC استفاده می شود تا reuse distance هر خط cache تخمین زده شود. این سیاست سه جز اصلی دارد:

۱. Sample Cache
۲. Reuse Distance Predictor(RDP)
۳. ETA Counter

Sample Cache

به طور خلاصه این جز reuse distance های قبلی را برای train کردن RDP استفاده می کند. برای این کار یک تاریخچه با اندازه ۸ برابر نمونه برداشته شده دارد که دسترسی های منحصر به فرد گذشته cache را نگه می دارد.

RDP

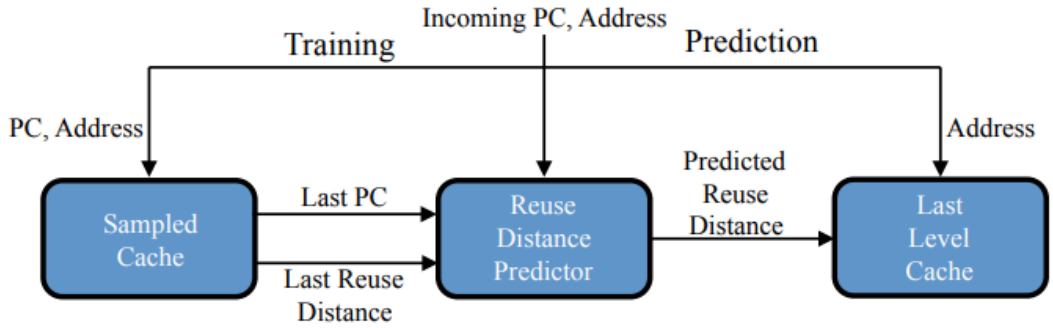
یک پیش بینی کننده بر اساس PC است که reuse distance را برای ورودی هایی که توسط PCوارد می شوند، یاد می گیرد. هر ورودی در RDP، ۰ مقداردهی می شود و با استفاده مجدد هر خط در sample cache ورودی RDP مربوط به PC که آخرین بار به خط دسترسی داشته، با استفاده از reuse distance مشاهده شده به روز می شود. چون مقدار reuse distance برای PC ممکن است نوسان کند، از پارامتری به اسم temporal difference learning (مقادیر پیش بینی شده را به روز می کند به عنوان یک تابع خطی از تفاوت بین مقادیر پیش بینی شده و مشاهده شده) برای آموزش RDP استفاده می کند .
به طور کلی ورودی های RDP به این صورت train می شود که اگر reduce distance جدید از مقدار قبلی بزرگتر بود مقدار ورودی با W جمع زده می شود و اگر کوچکتر بود، مقدار ورودی منهای w می شود. همچنین اگر وجود نداشت، ورودی به عنوان reuse distance ست می شود.

$$diff = absolute\ difference\ between\ the\ previous\ entry\ and\ the\ new\ reuse\ distance$$

$$w = min(1, \frac{diff}{16})$$

ETA counters

در نهایت خود cache ، ETA را برای هر خط حفظ می کند. پس از درج در حافظه نهان، فاصله استفاده مجدد پیش بینی شده خط به یک ETA تبدیل می شود و این مقادیر ETA برای تصمیم کیری برای خروج از حافظه پنهان آینده استفاده می شود. درج های cache که پیش بینی می شود مقادیر ETA بزرگتر از هر خط موجود در مجموعه داشته باشند، دور زده می شوند، به این معنی که در حافظه پنهان درج نمی شوند.



Estimated Time Remaining (ETR)

برای کاهش هزینه نگهداری time stamp های دقیق ETA برای هر خط cache، از یک مقدار کوچکتر اما منطقاً معادل، به نام Esti-mated Time Remaining (ETR) استفاده می‌شود. مقدار ETR با predicted reuse distance خط مقدار دهی اولیه می‌شود و هر بار که به خط دیگری در مجموعه دسترسی پیدا می‌شود، کاهش می‌یابد. با کاهش تفاوت بین زمان حال و ETA پیش بینی شده یک خط cache، ETR آن خط نیز کاهش می یابد. بنابراین، ترتیب نسبی ETR ها دقیقاً مشابه ترتیب نسبی ETA ها است.

Handling Imprecise Predictions

گاهی اوقات، سیستم ممکن است reuse distance یک خط را دست‌کم بگیرد و باعث شود مقدار شمارنده ETR آن بدون مشاهده reuse به ۰ برسد. اگر مقدار شمارنده در ۰ اشباع شود، این خط همیشه بالاترین اولویت ذخیره سازی را حفظ می کند که نامطلوب است. برعکس، اگر به چنین خطوطی یک مقدار ETR بی نهایت داده شود، آنگاه خطوطی که ETA آنها فقط با مقدار کمی دست کم گرفته شده است، بلافاصله خارج می شوند، که این نیز نامطلوب است. برای رسیدگی به این پیش‌بینی‌های غیردقیق، سیستم پس از رسیدن به مقدار ۰، شمارنده‌های ETR را کاهش می‌دهد. به عنوان مثال، ETR counter با مقدار 4- نشان می دهد که خط با ۴ دسترسی مجموعه ای از ETA خود فراتر رفته است. پس از اخراج، این سیاست، خطی را با بزرگترین مقدار مطلق ETR، که دورترین خط از ETA آن است، خارج می کند. بنابراین، خط اخراج شده یا پیش‌بینی می‌شود که در آینده بیشتر مورد استفاده مجدد قرار گیرد یا در گذشته پیش‌بینی می‌شود که در دورترین فاصله‌ها مورد استفاده مجدد قرار گیرد. پیوندها با بیرون کردن خطوط با ETR منفی به نفع خطوط با ETR مثبت شکسته می شوند.

بخش امتیازی

در این بخش ابتدا ۱۰ برنامه محک را به صورت رندم generate کردیم، برای این که بتوانیم این ۱۰ برنامه را با استفاده از ۴ سیاست مختلف روی gem5 شبیه سازی کنیم، باید مراحل زیر را انجام دهیم تا قابلیت سیاست حافظه نهان اضافه شود :

ابتدا باید قابلیت سیاست جایگزینی را اضافه کنیم، پس در فایل configs/common/ObjectList.py خط زیر را اضافه می کنیم:

repl-list = ObjectList(getattr(m5.objects , ”BaseReplacementPolicy”, None)) سپس در فایل configs/common/Options.py به تابع addNoISAOptions باید کد زیر را اضافه کنیم:

parser.add-argument(”–l2-repl”, default=”LRURP”, choices=ObjectList.repl-list.get-names(), help = ”replacement policy for l2”)

سپس در فایل configs/common/ CacheConfig.py به تابع get-cache-opts- کد زیر را اضافه می کنیم:

replacement-policy-attr = f”level-repl” if hasattr(options , replacement-policy-attr): opts[”replacement policy”] = ObjectList.repl-list.get(getattr(options ,replacement policy-attr))()

حال اگر در هنگام شبیه سازی دستور l2-repl- را بزنیم می توانیم نوع سیاست جایگزینی را مشخص کنیم و شبیه سازی را طبق آن انجام دهیم. در اینجا نیازی به ساخت فایل کانفیگ نیست و می توان از فایل پیش فرضی که در آدرس configs/deprecated/example/se.py است، استفاده کنیم.

حال از یک اسکریپت پایتون برای اجرای این برنامه ها با سیاست های مختلف استفاده می کنیم به این صورت که ابتدا در یک حلقه تمامی فایل های محک را به فایل باینری تبدیل می کنیم به کمک دستور زیر:

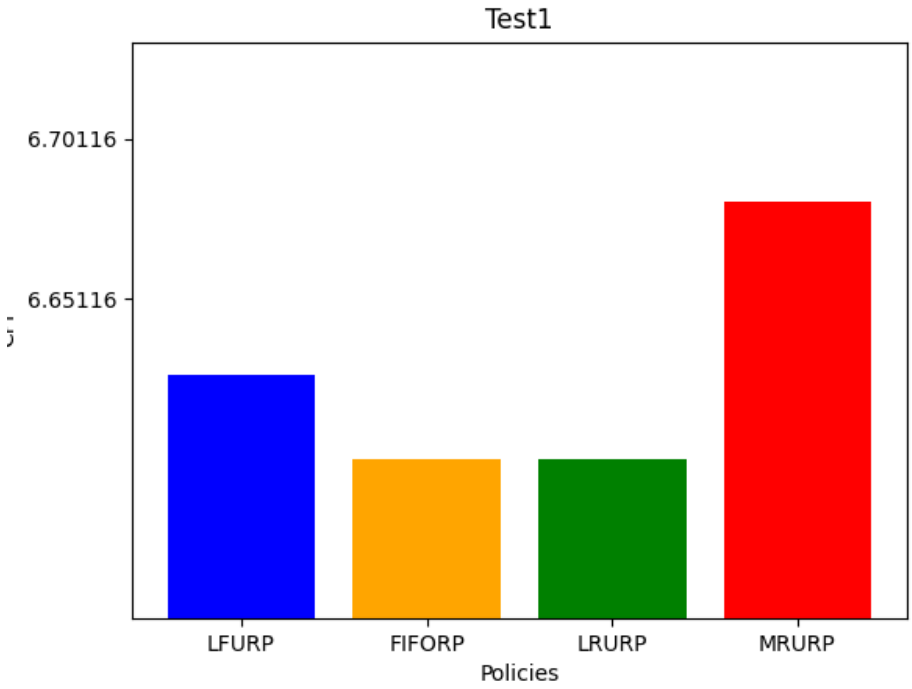
gcc -o tests/test tests/test.c

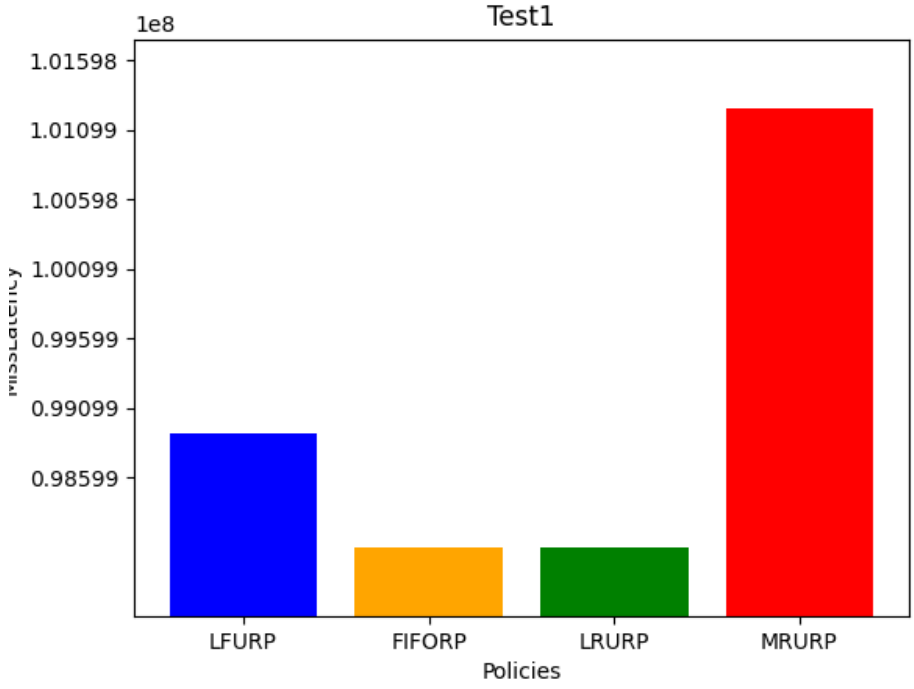
سپس تمام سیاست های مورد نظر را در یک آرایه قرار می دهیم و با زدن حلقه روی این فایل های باینری و سیاست ها تمام فایل ها را روی سیاست های مختلف اجرا می کنیم. در اصل از دستور زیر استفاده شده:

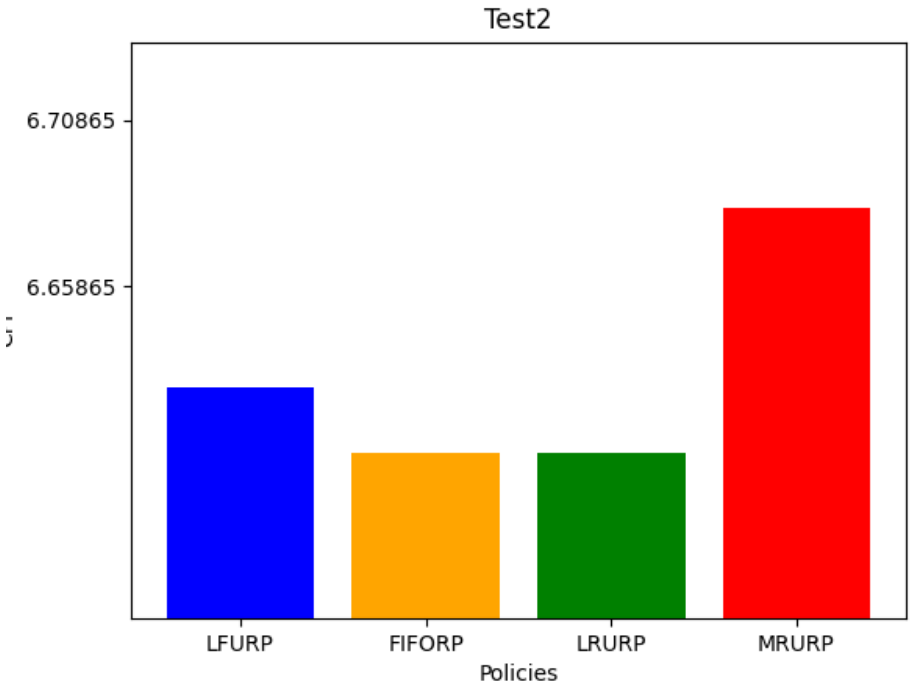
./build/X86/gem5.opt configs/deprecated/example/se.py -c tests/test –caches –l2cache –l2-size=4kB –mem-type=DDR4-2400-16x4 –cacheline-size 128 –cpu-type=TimingSimpleCPU –l2-repl=LRURP

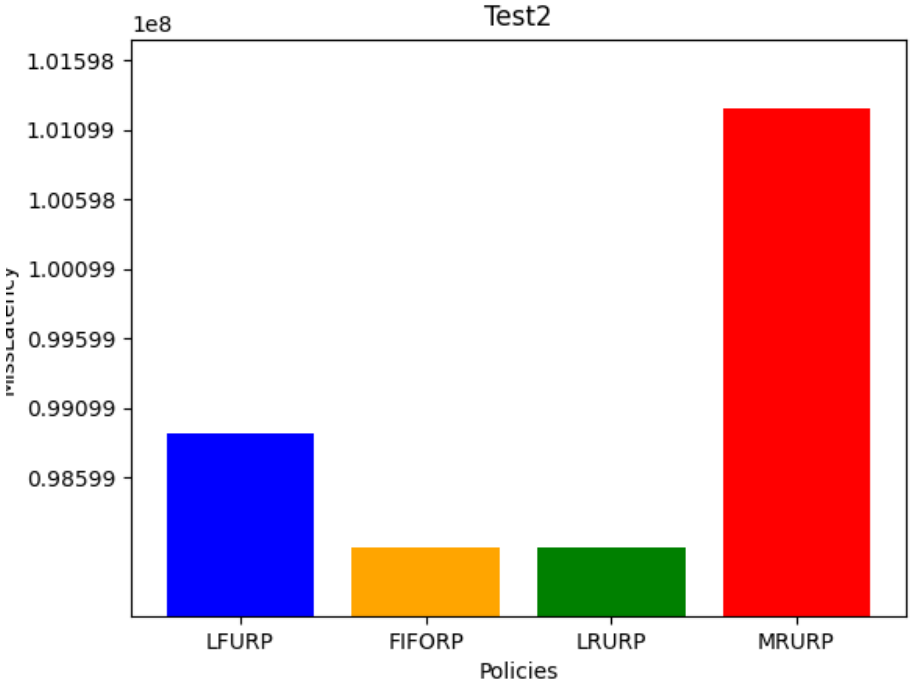
حال اطلاعات شبیه سازی این برنامه ها که در فایل stats.txt ذخیره شده است را داریم و اطلاعاتی مثل CPI و miss rate را از آن ها خوانده و در فایل csv ذخیره می کنیم. سپس با استفاده از این اطلاعات نمودار های مربوط به هر برنامه محک را با تفکیک سیاست ها رسم می کنیم.

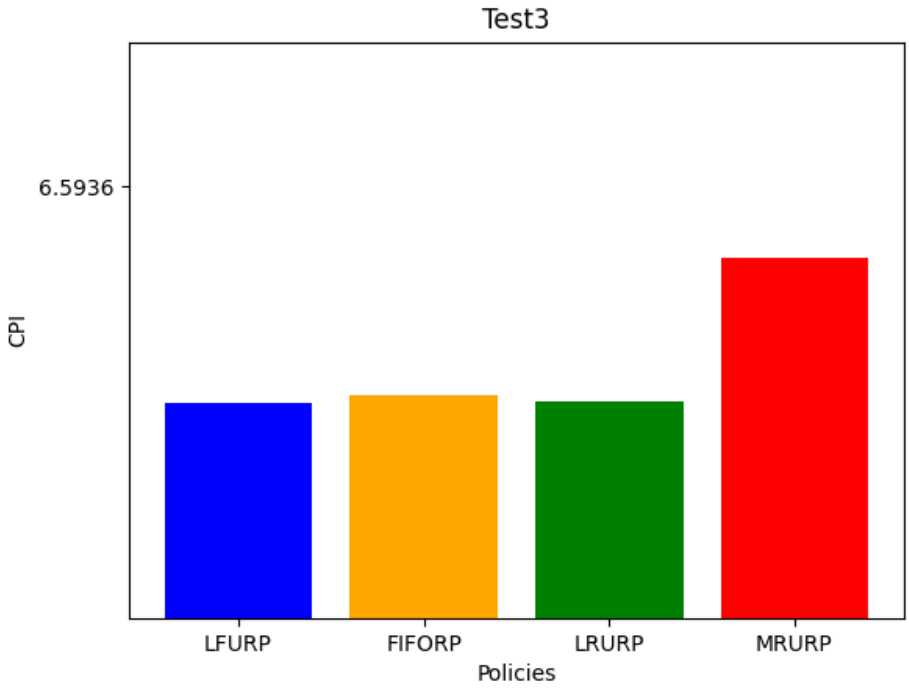
```python
def compile_c_program(source_file, output_file):
    try:
        subprocess.run(['gcc', '-o', output_file, source_file], check=True)
        print(f"Compilation successful. Binary created at {output_file}")
    except subprocess.CalledProcessError as e:
        print(f"Compilation failed: {e}")

def run_gem5_simulation(binary_file, repl_policy, test_number):
    try:
        gem5_command = [
            './build/X86/gem5.opt', 'configs/deprecated/example/se.py',
            '-c', binary_file,
            '--caches', '--l2cache',
            '--l2_size=4kB',
            '--mem-type=DDR4_2400_16x4',
            '--cacheline_size', '128',
            '--cpu-type=TimingSimpleCPU',
            f'--l2_repl={repl_policy}'
       ]

        print(f"Running gem5 command: {' '.join(gem5_command)}")
        result = subprocess.run(gem5_command, check=True, capture_output=True, text=True)
        print(f"Simulation successful for {binary_file} with replacement policy {repl_policy}")
        print(f"Output: {result.stdout}")
        print(f"Errors: {result.stderr}")
        stats_file = 'm5out/stats.txt'
        new_stats_file = f'm5out/stats_test{test_number}_{repl_policy}.txt'
        if os.path.exists(stats_file):
            os.rename(stats_file, new_stats_file)
            print(f"Renamed {stats_file} to {new_stats_file}")
            return new_stats_file
        else:
            print(f"{stats_file} does not exist")
            return None
    except subprocess.CalledProcessError as e:
        print(f"Simulation failed for {binary_file} with replacement policy {repl_policy}:
{e}")   print(f"Output: {e.stdout}")
        print(f"Errors: {e.stderr}")
        return None

def extract_stats(stats_file):
    stats = {}
    if stats_file and os.path.exists(stats_file):
        with open(stats_file, 'r') as file:
            for line in file:
                if 'system.cpu.cpi' in line:
                    stats['system.cpu.cpi'] = float(line.split()[1])
                elif 'system.cpu.dcache.demandMissLatency::total' in line:
                    stats['system.cpu.dcache.demandMissLatency::total'] = float(line.split()
[1])return stats

def save_stats_to_csv(stats, test_number, repl_policy):
    if not os.path.exists('csvFiles'):
        os.makedirs('csvFiles')

    csv_file = f'csvFiles/csv{test_number}{repl_policy}.csv'
    with open(csv_file, 'w', newline='') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(['Metric', 'Value'])
        for key, value in stats.items():
            writer.writerow([key, value])
    print(f"Saved stats to {csv_file}")

replacement_policies = ['LRURP', 'FIFORP', 'LFURP', 'MRURP']

for i in range(1, 11):
    source_file = f'tests/New folder/test{i}.c'
    output_file = f'tests/New folder/test{i}'
    compile_c_program(source_file, output_file)

    for repl_policy in replacement_policies:
        stats_file = run_gem5_simulation(output_file, repl_policy, i)
        stats = extract_stats(stats_file)
        save_stats_to_csv(stats, i, repl_policy)
```

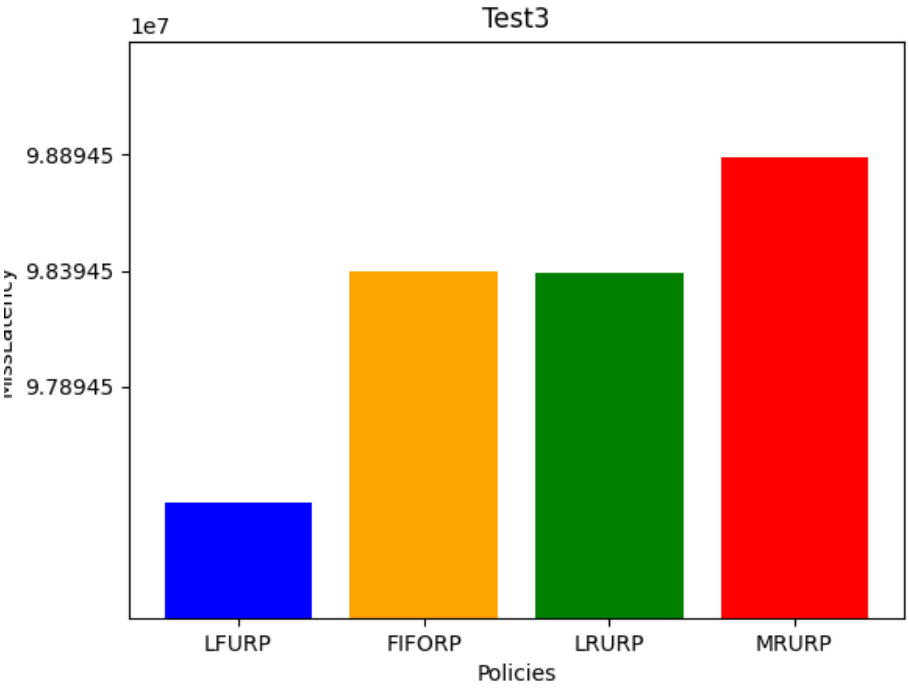

Test1
6.70116
6.65116
LFURP
FIFORP
LRURP
MRURP
Policies

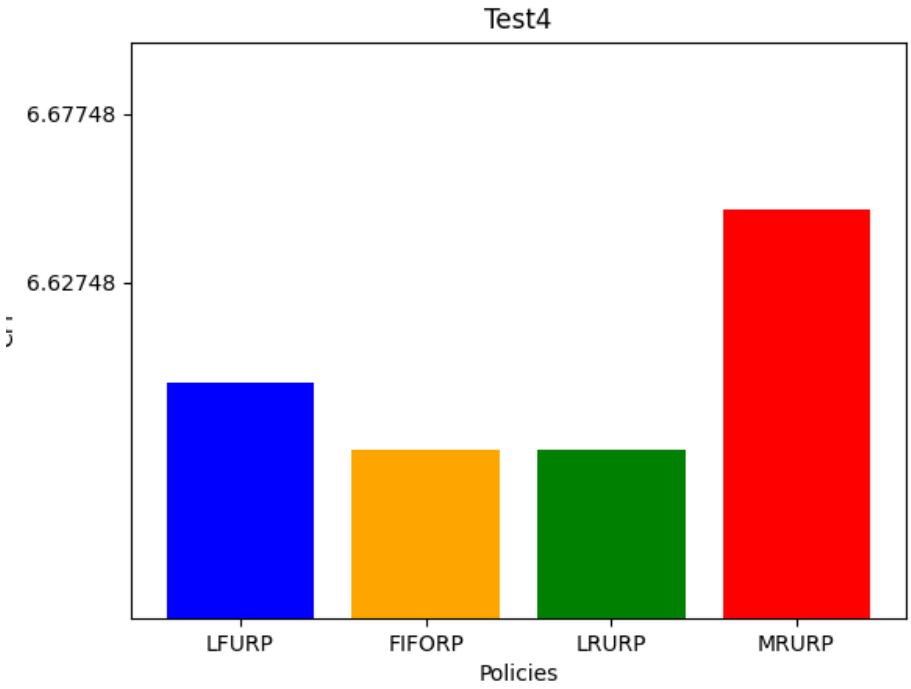

Test1
1e8
1.01598
1.01099
1.00598
1.00099
0.99599
0.99099
0.98599
LFURP
FIFORP
LRURP
MRURP
Policies

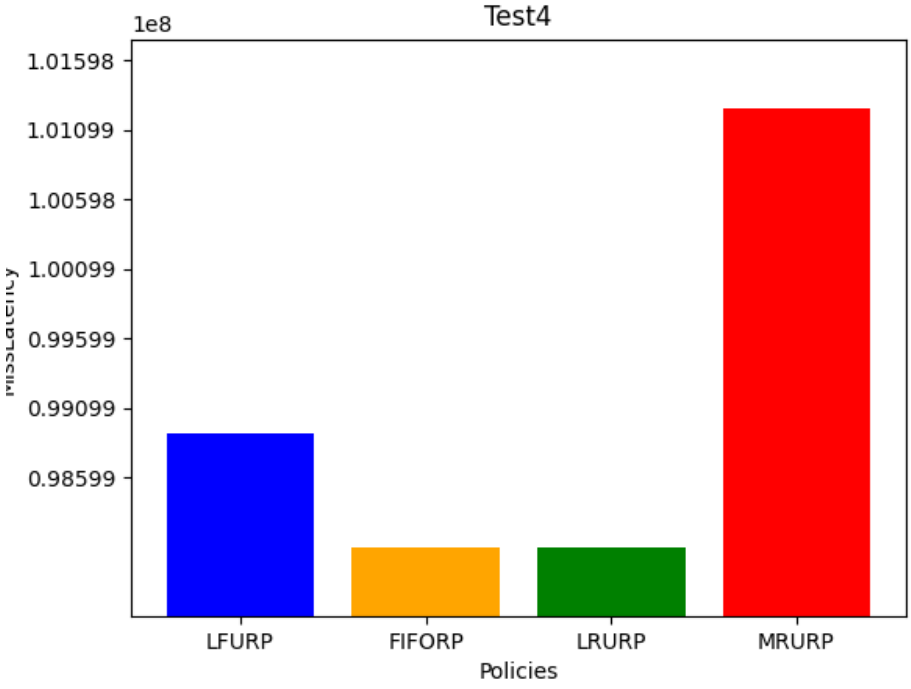

Test2
6.70865
6.65865
LFURP
FIFORP
LRURP
MRURP
Policies

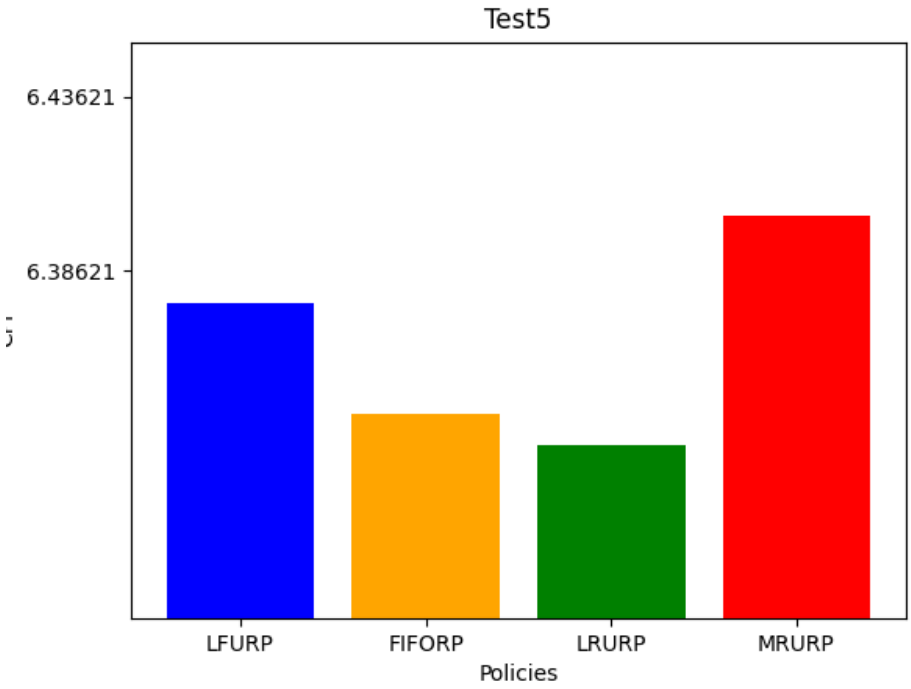

Test2
1e8
1.01598
1.01099
1.00598
1.00099
0.99599
0.99099
0.98599
LFURP
FIFORP
LRURP
MRURP
Policies

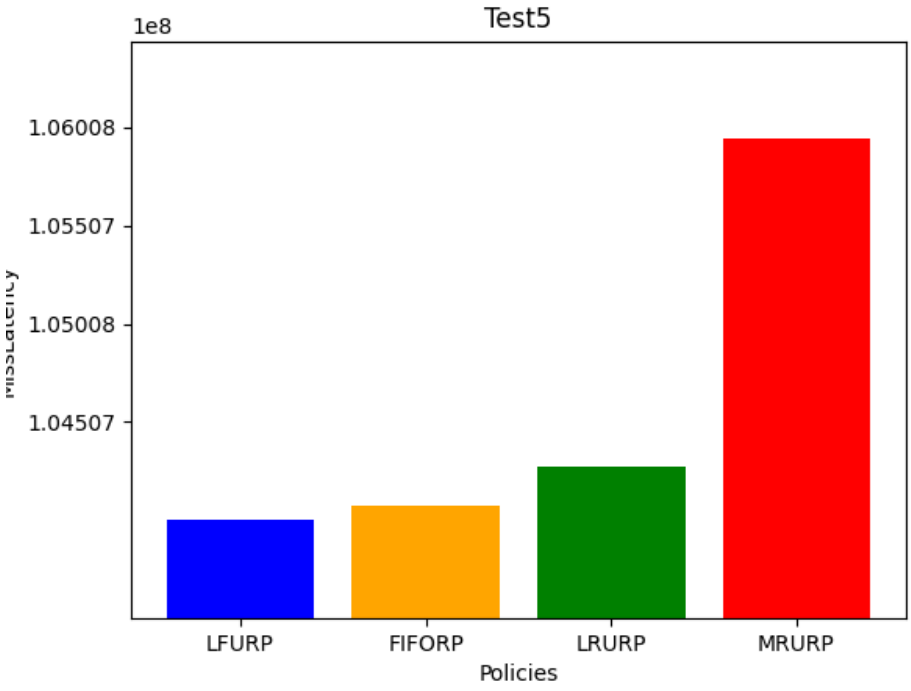

Test3
6.5936
CPI
LFURP
FIFORP
LRURP
MRURP
Policies

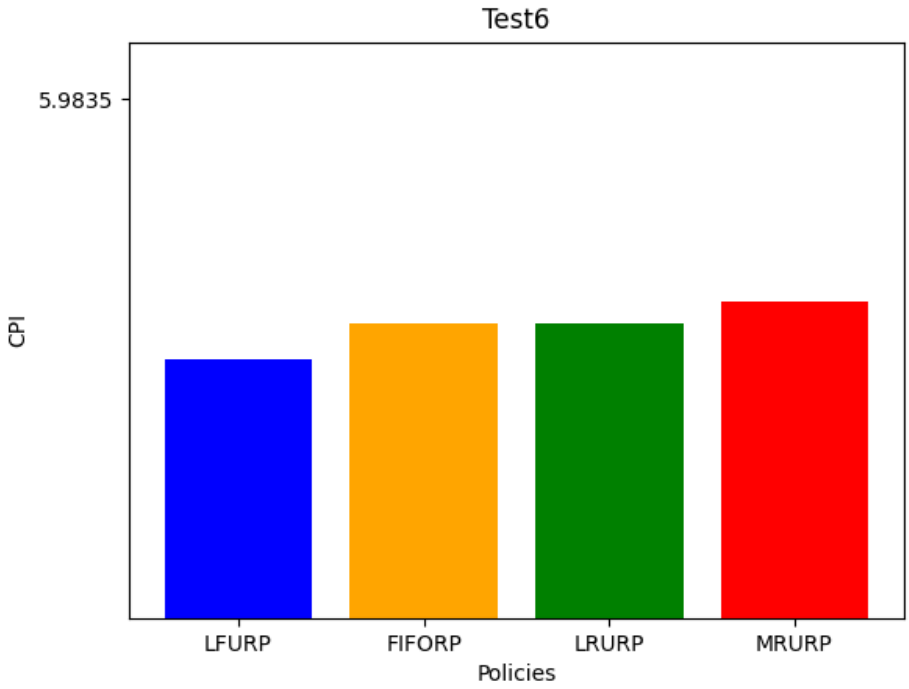

Test3
1e7
9.88945
9.83945
9.78945
LFURP
FIFORP
LRURP
MRURP
Policies

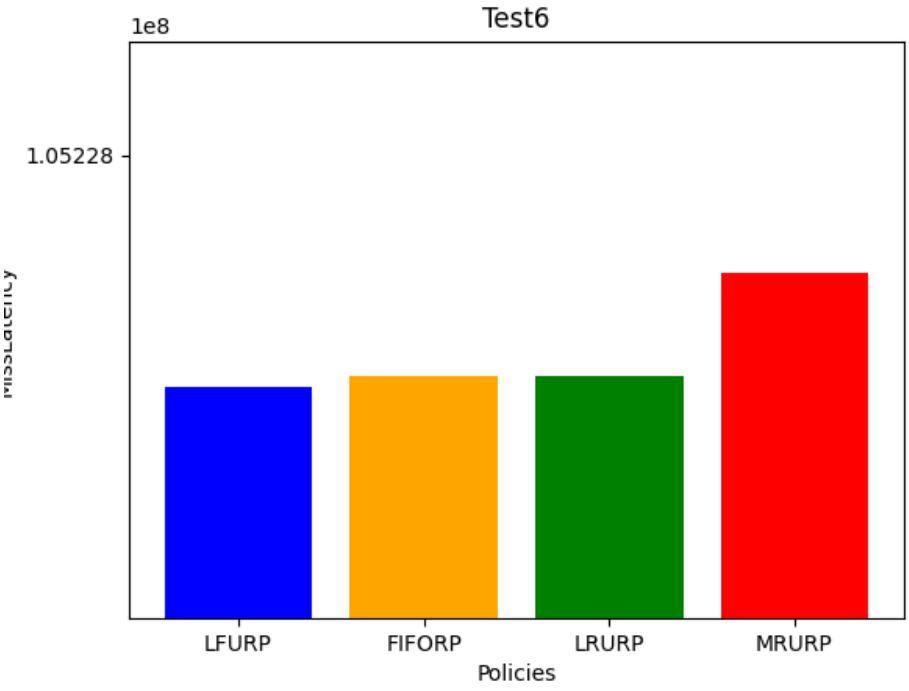

Test4
6.67748
6.62748
LFURP
FIFORP
LRURP
MRURP
Policies

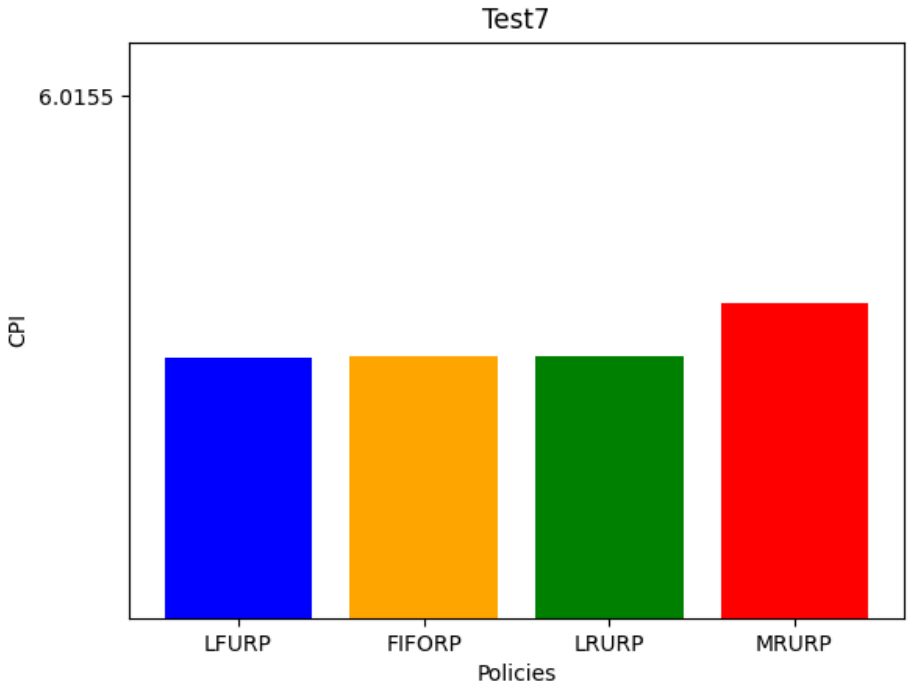

Test4
1e8
1.01598
1.01099
1.00598
1.00099
0.99599
0.99099
0.98599
LFURP
FIFORP
LRURP
MRURP
Policies

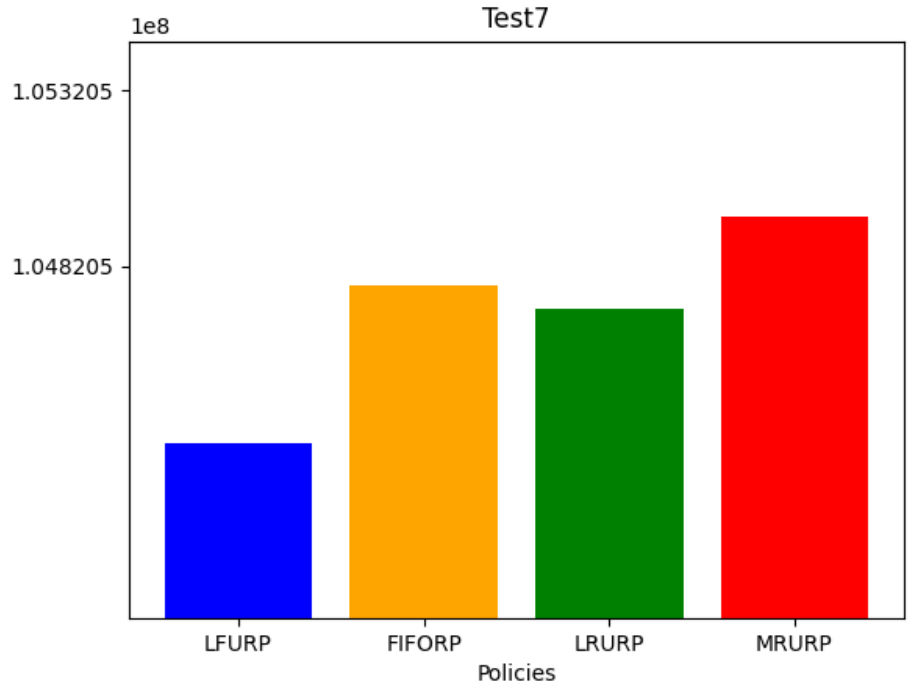

Test5
6.43621
6.38621
LFURP
FIFORP
LRURP
MRURP
Policies

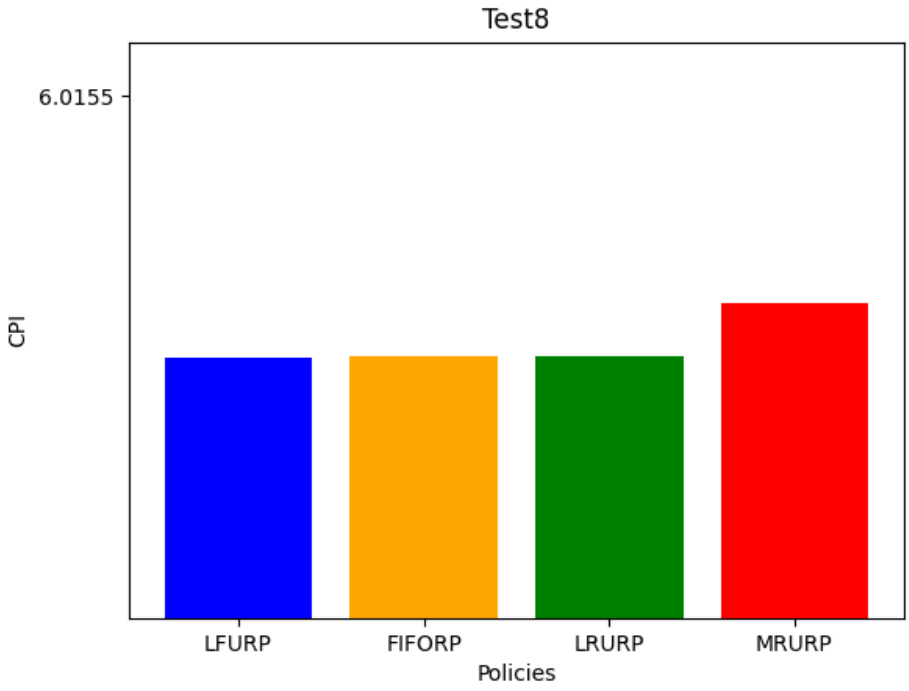

Test5
1e8
1.06008
1.05507
1.05008
1.04507
LFURP
FIFORP
LRURP
MRURP
Policies

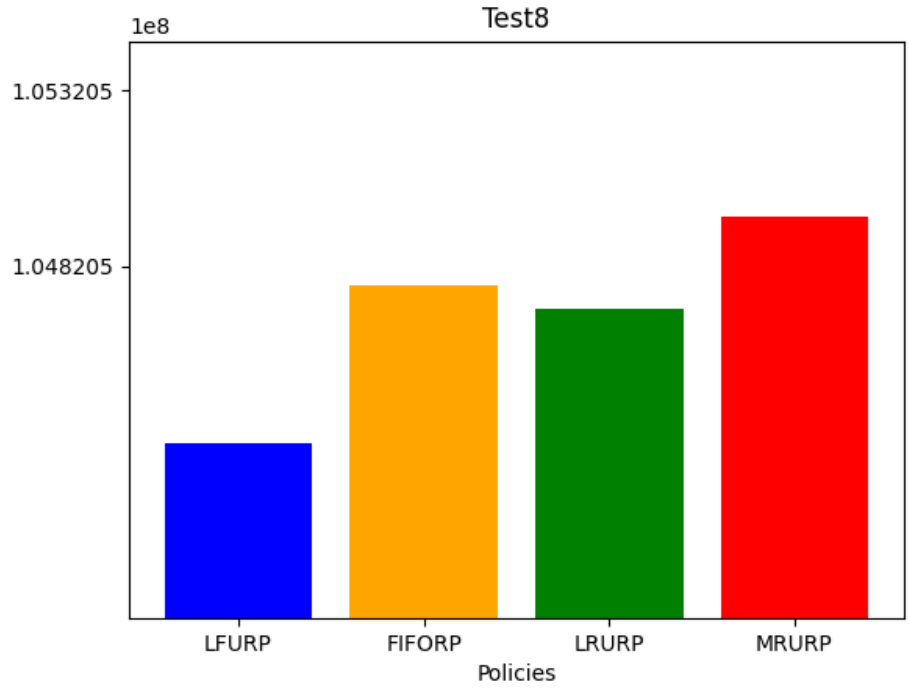

Test6
5.9835
CPI
LFURP
FIFORP
LRURP
MRURP
Policies

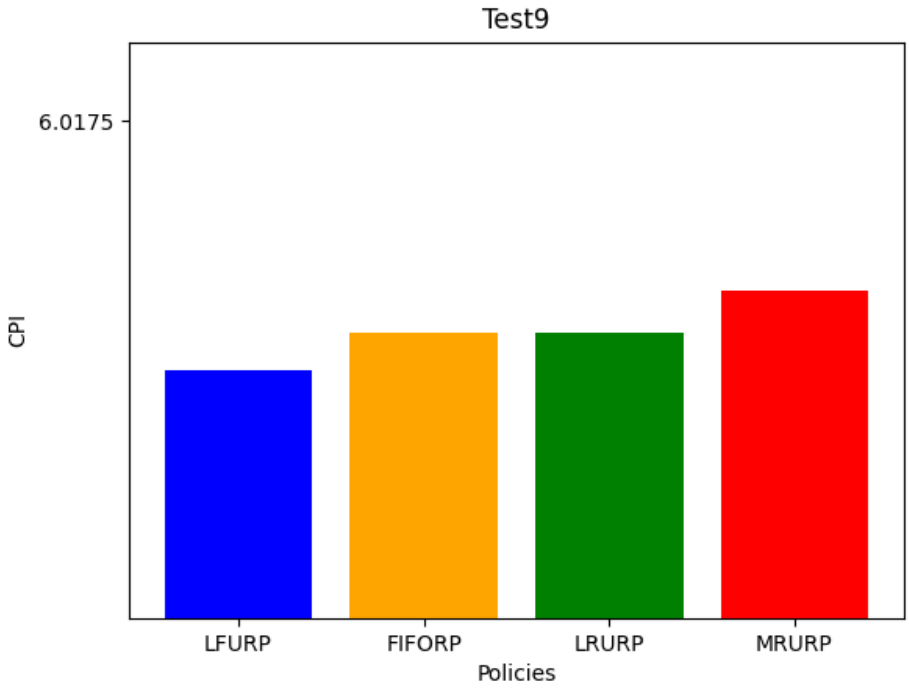

Test6
1e8
1.05228
LFURP
FIFORP
LRURP
MRURP
Policies

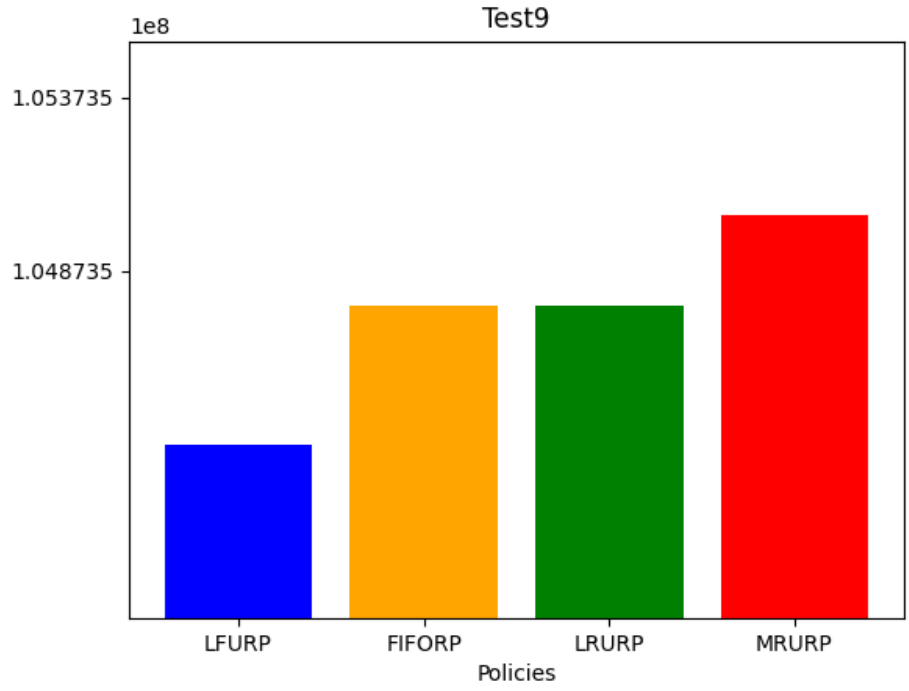

Test7
6.0155
CPI
LFURP
FIFORP
LRURP
MRURP
Policies

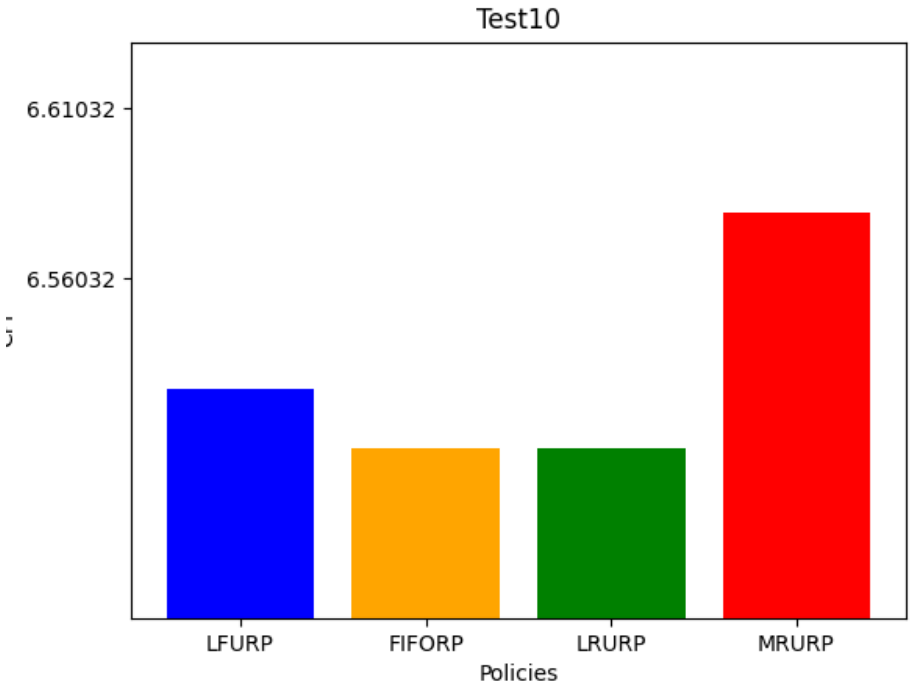

Test7
1e8
1.053205
1.048205
LFURP
FIFORP
LRURP
MRURP
Policies

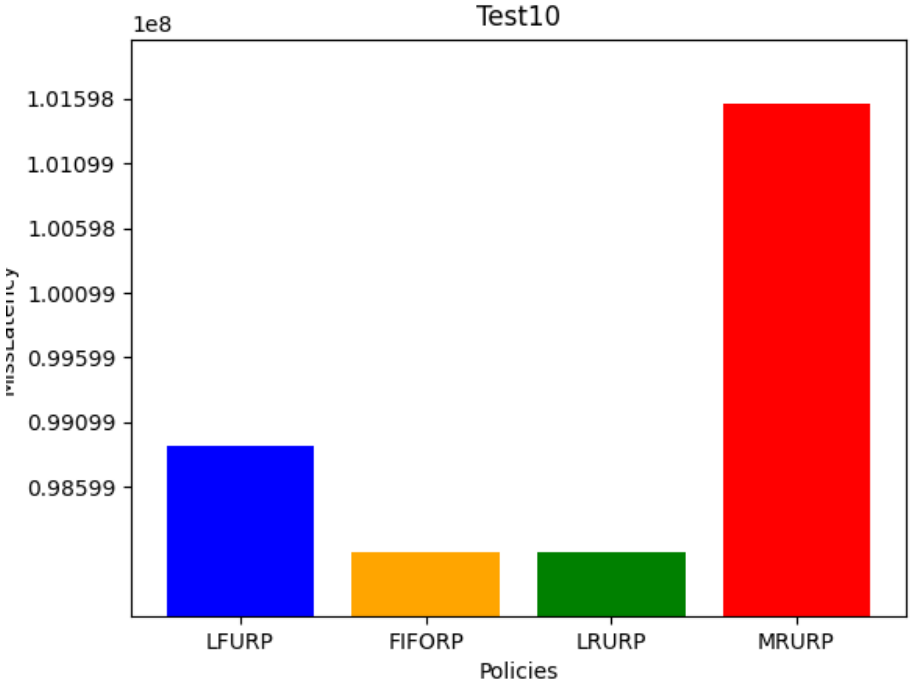

Test8
6.0155
CPI
LFURP
FIFORP
LRURP
MRURP
Policies


Test8
1e8
1.053205
1.048205
LFURP
FIFORP
LRURP
MRURP
Policies


Test9
6.0175
CPI
LFURP
FIFORP
LRURP
MRURP
Policies


Test9
1e8
1.053735
1.048735
LFURP
FIFORP
LRURP
MRURP
Policies


Test10
6.61032
6.56032
LFURP
FIFORP
LRURP
MRURP
Policies


Test10
1e8
1.01598
1.01099
1.00598
1.00099
0.99599
0.99099
0.98599
LFURP
FIFORP
LRURP
MRURP
Policies