Benchmark

August 18, 2024

1 Benchmark TSP48

1.1 Setup TSP48

[2]:
```
rEval=lcl_rust.Evaluation.tsp_from_dist_matrix("../data/distanceMatrix")

rMove0=lcl_rust.MoveType.swap_tsp()
rMove1=lcl_rust.MoveType.reverse()
rProblemSwap=lcl_rust.Problem.array_problem(rMove0,rEval)
rProblemReverse=lcl_rust.Problem.array_problem(rMove1,rEval)

#simulated annealing
rMinTemp=lcl_rust.Termination.min_temp(100)
rCooling=lcl_rust.Cooling.geometric_cooling(0.95)
rIter=lcl_rust.IterationsPerTemp.cnst_iter_temp(1500)
rSimAnn=lcl_rust.LocalSearch.
  ↪simulated_annealing(2000,True,rProblemSwap,rMinTemp,rCooling,rIter)

#steepest descent
rMax5=lcl_rust.Termination.max_sec(5)
rSteepestDescent=lcl_rust.LocalSearch.steepest_descent(True,rProblemSwap,rMax5)

#tabu search
rTabuSearch=lcl_rust.LocalSearch.tabu_search(True,rProblemSwap,rMax5)

#vns
rMoveVNS=lcl_rust.MoveType.multi_neighbor([rMove0,rMove1])
rProblemVNS=lcl_rust.Problem.array_problem(rMoveVNS,rEval)
rVNS=lcl_rust.LocalSearch.vns(True,rProblemVNS,rMax5)

pData=numpy.loadtxt("../data/distanceMatrix",dtype=float)

#Moves
pMoveSwap = TspArraySwap(pData.shape[0])
pMoveReverse=ArrayReverseOrder(pData.shape[0])
#Eval
pEvalTSPSwap=TspEvaluationFunction(pData,pMoveSwap)
pEvalTSPReverse=TspEvaluationFunction(pData,pMoveReverse)
```

```
# termination
pTermMax5=MaxSecondsTerminationCriterion(5)
pTermMax5_2=MaxSecondsTerminationCriterion(5)
#Problem
pProblemSwap=ArrayProblem(pEvalTSPSwap,pMoveSwap,pData.shape[0])
pProblemReverse=ArrayProblem(pEvalTSPReverse,pMoveReverse,pData.shape[0])

# local search
pCooling = GeometricCoolingFunction(0.95)
pIter = CnstIterationsTempFunction(1500)
pTerminationTemp = MinTemperatureTerminationCriterion(100)
pSimAnn = SimulatedAnnealing(None, pTerminationTemp, pCooling, pIter,␣
 ↪benchmarking=True,logging=False)
pSteepestDescent=SteepestDescent(None,termination_criterion=pTermMax5,␣
 ↪benchmarking=True,logging=False)

pDiffState =SumDiffState()
pTabuSearch=TabuSearch(None,pTermMax5_2,pDiffState,␣
 ↪benchmarking=True,logging=False)

#vns
pMax5VNS = MaxSecondsTerminationCriterion(5)
pMove0 = TspArraySwap(pData.shape[0])
pMove1 = ArrayReverseOrder(pData.shape[0])
pMoveVNS = MultiNeighbourhood([pMove0,pMove1])

pEvalVNS=TspEvaluationFunction(pData,pMoveVNS)
pProblemVNS=ArrayProblem(pEvalVNS,pMoveVNS,pData.shape[0])
pVNS = VariableNeighbourhood(pProblemVNS,logging=False, benchmarking=True)
```

1.2 Run tests

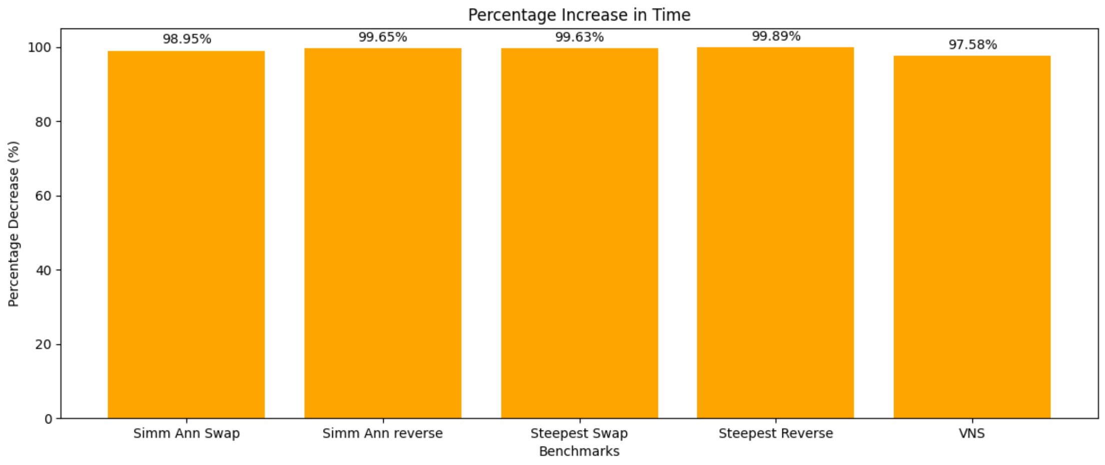
[3]:
```
rustVNS=lcl_rust.benchmark([rVNS],[rProblemVNS],runs=10)
pythonVNS=pBenchmark([pVNS],[pProblemVNS],runs=10)

rustBenchRes=lcl_rust.benchmark(
    [rSimAnn,rSteepestDescent,rTabuSearch],
    [rProblemSwap,rProblemReverse],runs=10)
pythonbenchRes=pBenchmark(
    [pSimAnn,pSteepestDescent,pTabuSearch],
    [pProblemSwap,pProblemReverse],runs=10)
```

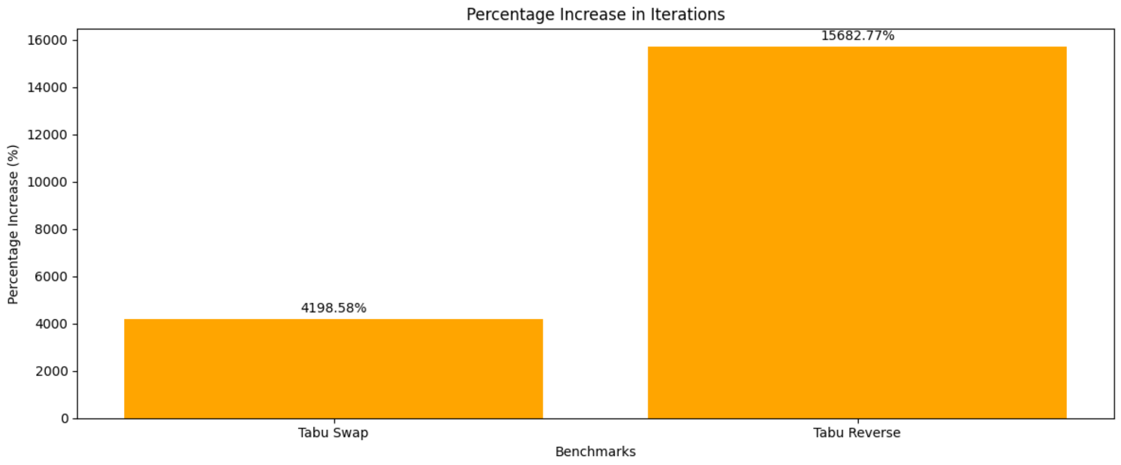
1.3 Crunch data

[6]:
```
rTimeAvg=[]
pTimeAvg=[]
rIterationAvg=[]
pIterationAvg=[]
for alg in rustBenchRes:
    for prob in alg:
        rTime=[t[-1][0]/(10**9) for t in prob]
        rIteration=[t[-1][3] for t in prob]
        rTimeAvg.append(sum(rTime)/len(rTime))
        rIterationAvg.append(sum(rIteration)/len(rIteration))
for alg in pythonbenchRes:
    for prob in alg:
        pTime=[seed.data.time[-1] for seed in prob]
        pIteration=[seed.data.iteration[-1] for seed in prob]
        pTimeAvg.append(sum(pTime)/len(pTime))
        pIterationAvg.append(sum(pIteration)/len(pIteration))

for alg in rustVNS:
    for prob in alg:
        rTime=[t[-1][0]/(10**9) for t in prob]
        rIteration=[t[-1][3] for t in prob]
        rTimeAvg.append(sum(rTime)/len(rTime))
        rIterationAvg.append(sum(rIteration)/len(rIteration))
for alg in pythonVNS:
    for prob in alg:
        pTime=[seed.data.time[-1] for seed in prob]
        pIteration=[seed.data.iteration[-1] for seed in prob]
        pTimeAvg.append(sum(pTime)/len(pTime))
        pIterationAvg.append(sum(pIteration)/len(pIteration))

pTimeAvg=[float(p) for p in pTimeAvg]
pIterationAvg=[float(p) for p in pIterationAvg]
time_percentage_increase=[abs(t2-t1)/t1 *100 if t1!=0 else 0 for t1,t2 in␣
 ↪zip(pTimeAvg,rTimeAvg)]
iteration_percentage_increase =[(t2-t1)/t1 *100 if t1!=0 else 0 for t1,t2 in␣
 ↪zip(pIterationAvg,rIterationAvg)]
print(time_percentage_increase)
print(iteration_percentage_increase)
del time_percentage_increase[-3:-1]
iteration_percentage_increase=iteration_percentage_increase[-3:-1]
```

```
[98.9474984801982, 99.65156250226977, 99.63102635607775, 99.8896647539281,
0.11407887463899045, 0.3487769961539119, 97.57906234942548]
[0.0, 0.0, 0.0, 0.0, 4198.5773614915615, 15682.769857433808,
-30.589849108367627]
```

[5]:
```
# Create figure and subplots
fig, (ax1,ax2) = plt.subplots(2, 1, figsize=(12, 10))

# Plot time percentage increase
benchmarksTime=["Simm Ann Swap","Simm Ann reverse","Steepest Swap","Steepest␣
 ↪Reverse","VNS"]
barsTime=ax1.bar(numpy.arange(len(benchmarksTime)), time_percentage_increase,␣
 ↪color='orange')
ax1.set_title('Percentage Increase in Time')
ax1.set_xlabel('Benchmarks')
ax1.set_ylabel('Percentage Decrease (%)')
ax1.set_xticks(numpy.arange(len(benchmarksTime)))
ax1.set_xticklabels(benchmarksTime)

for bar in barsTime:
    height = bar.get_height()
    ax1.annotate(f'{height:.2f}%',
                 xy=(bar.get_x() + bar.get_width() / 2, height),
                 xytext=(0, 3),  # 3 points vertical offset
                 textcoords="offset points",
                 ha='center', va='bottom')

# Plot iteration percentage increase
benchmarksIterations=["Tabu Swap","Tabu Reverse"]
barsIteration=ax2.bar(numpy.arange(len(benchmarksIterations)),␣
 ↪iteration_percentage_increase, color='orange')
ax2.set_title('Percentage Increase in Iterations')
ax2.set_xlabel('Benchmarks')
ax2.set_ylabel('Percentage Increase (%)')
ax2.set_xticks(numpy.arange(len(benchmarksIterations)))
ax2.set_xticklabels(benchmarksIterations)
for bar in barsIteration:
    height = bar.get_height()
    ax2.annotate(f'{height:.2f}%',
                 xy=(bar.get_x() + bar.get_width() / 2, height),
                 xytext=(0, 3),  # 3 points vertical offset
                 textcoords="offset points",
                 ha='center', va='bottom')
# Adjust layout
plt.tight_layout()

# Show the plot
plt.show()
```


Percentage Increase in Time
Percentage Decrease (%)
0
20
40
60
80
100
98.95%
99.65%
99.63%
99.89%
97.58%
Simm Ann Swap
Simm Ann reverse
Steepest Swap
Steepest Reverse
VNS
Benchmarks


Percentage Increase in Iterations
Percentage Increase (%)
0
2000
4000
6000
8000
10000
12000
14000
16000
4198.58%
15682.77%
Tabu Swap
Tabu Reverse
Benchmarks