LINE Solver Cheatsheet (Python)

http://line-solver.sf.net

Node Types

Cache	Caching station
ClassSwitch	Class switching
Delay	Infinite server
Fork	Forks jobs
Join	Joins jobs
Logger	Trace logging
Place	Petri net place
Queue	Queueing station
Router	Routing node
Sink	Job departures (open)
Source	Job arrivals (open)
Transition	Petri net transition

Job Classes

```
ClosedClass(model, 'name', N, refStat)
OpenClass(model, 'name')
SelfLoopingClass(model, 'name', N, ref)
DisabledClass(model, 'name')
```

OpenSignal, ClosedSignal (catastrophes)

Distributions

APH(α, T)	Acyclic PH
Cox2(μ, ϕ)	Two-phase Coxian
Coxian(μ, ϕ)	Coxian
Det(v)	Deterministic
Disabled	None
Erlang(λ, k)	Erlang-k
Exp(λ)	Exponential
Gamma(α, β)	Gamma
HyperExp(p, λ_1, λ_2)	Hyper-exp
Immediate	Zero service time
Lognormal(μ, σ)	Log-normal
Normal(μ, σ)	Normal
Pareto(α, k)	Pareto
PH(α, T)	Phase-type
Uniform(a, b)	Uniform
Weibull(α, β)	Weibull

Discrete

Bernoulli(p)	Bernoulli
Binomial(p, n)	Binomial
DiscreteUniform(a, b)	Discr. uniform
Geometric(p)	Geometric
Poisson(λ)	Poisson
Zipf(s, n)	Popularity

Correlated / trace-driven

BMAP(D0..Dk)	Batch MAP
NHPP(bp,λ,cyclic)	Non-homog. Poisson
DMAP(D0, D1)	Discrete MAP
EmpiricalCDF(F)	Empirical CDF
MAP(D0, D1)	MAP
MarkedMAP	Marked MAP
ME(α, T)	Matrix-exp.
CME(mean, order)	Concentr. matrix-exp.
MMPP2(λ_1, λ_2, σ_1, σ_2)	MMPP(2)
RAP(H0, H1)	Rational AP
Replayer(trace)	Trace replay

Scheduling Strategies

DPS	Discriminatory PS
EDD	Earliest Due Date
EDF	Earliest Deadline First
EXT	External (source)
FB	Feedback (LAS)
FCFS	First-Come First-Served
FCFSPI	FCFS Preempt. Resume-idle
FCFSPR	FCFS Preemptive Resume
FSP	Fair Sojourn Protocol
GPS	Generalized PS
HOL	Head-of-Line Priority
INF	Infinite Servers
LCFS	Last-Come First-Served
LCFSPI	LCFS Preempt. Resume-idle
LCFSPR	LCFS Preemptive Resume
LEPT	Longest Expected PT
LJF	Longest Job First
LPS	Limited PS
LRPT	Longest Remaining PT
OI	Order Independent
PAS	Pass and Swap
POLLING	Polling (switchover)
PS	Processor Sharing
PSJF	Preemptive SJF
SEPT	Shortest Expected PT
SETF	Shortest Elapsed Time First
SIRO	Random
SJF	Shortest Job First
SRPT	Shortest Remaining PT

Solvers

AUTO	Auto-select best
BA	Bound analysis (closed)
CTMC	CTMC (exact, small)
ENV	Random environments
FLD	Fluid/mean-field ODEs
LDES	Discrete Event Sim
LN	Layered networks
MAM	Matrix Analytic Methods
MVA	Mean Value Analysis
NC	Normalizing Constant Methods
SSA	Stochastic simulation

Wrappers

JMT	JMT simulator
LQNS	LQNS interface
QNS	LQNS product-form solver

Solver Options

```
MVA(model, method='exact')
JMT(model, seed=123,
samples=10000)
SSA(model, samples=5000,
method='para')
LDES(model, MVA(model)) # warm start
```

Output Metrics

ArvR	Arrival rate
QLen	Mean queue length
ResidT	Residence time (total across visits)
RespT	Response time (per visit)
Tput	Throughput
Util	Server utilization
QLenVar	Queue-length variance
QLenSCV	Queue-length squared coeff. of vari
QLenSkew	Queue-length skewness
RespTVar	Response-time variance (FCFS only)
RespTSCV	Response-time SCV (FCFS only)
RespTSkew	Response-time skewness (FCFS only)

Moments (product form)

```
solver.getMomentTable()
# per class; order 2 by default
solver.getMomentTable(3)
# 1=means 2=+var 3=+skew
solver.getMomentChainTable(3)
# per chain
solver.getMomentStationTable(3)
# per station total
solver.getMomentStationTable(3,'lin')
# approx., for large N
```

order is a set: scalar k = up to k;
a vector, e.g. [2,3], = exactly those.
RespT$\star$ moments are FCFS-only
(NaN elsewhere); exact methods cost
$\prod_r (N_r+1)$, use 'lin' beyond that.

Analysis Methods

```
solver.avg_table()
solver.getAvgQLen(), getAvgUtil()
solver.getAvgRespT(), getAvgTput()
solver.getAvgSysRespT()
solver.getAvgSysTput()
solver.getAvgQLenChain()
```

Table Methods

getAvgTable	per station
getAvgChainTable	per chain
getAvgNodeTable	per node
getAvgNodeChainTable	node $\times$ chain
getAvgSysTable	system totals

Routing

```
model.link(Network.serial_routing(
n1,n2,n3))
n1.set_prob_routing(jobclass, n2, 0.7)
n1.set_routing(jobclass,
RoutingStrategy.RROBIN)
```

Multi-Server & Finite Capacity

```
queue.set_number_of_servers(k)
queue.set_capacity(bufferSize)
```

Class Switching

```
cs = ClassSwitch(model, 'CS')
P = [[0.3, 0.7], [0.5, 0.5]]
cs.set_class_switching_matrix(P)
```

Fork-Join Networks

```
fork = Fork(model, 'Fork')
join = Join(model, 'Join', fork)
```

Layered Networks

```
lqn = LayeredNetwork('name')
P = Processor(lqn,'P',1,
SchedStrategy.PS)
T = Task(lqn,'T',1,
SchedStrategy.REF).on(P)
E = Entry(lqn,'E').on(T)
A = Activity(lqn,'A').on(T) \
.bound_to(E).set_host_demand(Exp(1))
```

Caching

```
cache = Cache(model,'Cache',
nItems,cap,ReplacementStrategy.LRU)
cache.set_read(jobclass, Zipf(1.4, nItems))
# cache results
MVA(model).getAvgCacheTable() # hit/miss
MVA(model).getAvgItemTable() # per-item
```

ReplacementStrategy

CLIMB	Climb
FIFO	First-in first-out
HLRU	h-LRU
LRU	Least recently used
QLRU	q-LRU
RR	Random replacement
SFIFO	Static FIFO

Model Import/Export

```
model.jsimgView() # Open in JMT
save_model(model,'model.json')
model = load_model('model.json')
```

Routing Strategies

`JSQ`	Join shortest queue
`KCHOICES`	Power of k choices
`PROB`	Probabilistic routing
`RAND`	Random selection
`RROBIN`	Round-robin
`WRROBIN`	Weighted round-robin

Stochastic Petri Nets

```
p = Place(model, 'P1')
t = Transition(model, 'T1')
m = t.add_mode('Mode1')
t.set_number_of_servers(m, float('inf'))
t.set_distribution(m, Exp(4))
t.set_enabling_conditions(m, jobclass, p, 1)
t.set_inhibiting_conditions(m, jobclass, p, 3)
t.set_firing_outcome(m, jobclass, sink, 1)
```

Random Environments

```
env = Environment('R', 2)
env.add_stage(0,'Fast','operational',fastModel)
env.add_stage(1,'Slow','degraded',slowModel)
env.add_transition(0,1,Exp(0.5))
env.add_transition(1,0,Exp(1.0))
ENV(env, lambda m:  FLD(m)).getAvgTable()
```

Finite Capacity Regions

```
fcr = model.add_region([queue1, queue2])
fcr.set_global_max_jobs(10)
fcr.set_class_max_jobs(class1, 5)
fcr.set_drop_rule(class1, DropStrategy.DROP)
```

Load-Dependent Service

```
queue.set_load_dependence(np.minimum(np.arange(1,N+1),
c))
queue.set_class_dependence(lambda ni:  min(sum(ni),
c), c)
```

Impatience & Retrials

```
queue.set_patience(jobclass,
ImpatienceType.RENEGING, Exp(0.1))
queue.set_balking(jobclass, strategy, thresholds)
queue.set_retrial(jobclass, Exp(2), maxAttempts)
```

State Probabilities

```
solver.getProb(node) # marginal
solver.getProbAggr(node) # aggregate
solver.getProbSys() # joint
solver.getGenerator() # CTMC infin.
solver.getStateSpace()
```

Transient & Distributions

```
solver.getTranAvg() # transient means
solver.getCdfRespT() # response-time CDF
model.set_reward('Cost', lambda state:  ...)
solver.getAvgReward()
```