

Oct 06, 14 20:03

"csc710sbse: hw4: Witschey"

Page 1/13

```
##### Part 3 #####
2014-10-06 20:02:44.540058
```

```
Schaffer
```

```
5 -----
SimulatedAnnealer
era_length: 50, initialized: 2014-10-06 20:02:44.540153, iterations: 1000, log_eras: True, p_mutation: 0.333333333333, s
earcher: SimulatedAnnealer, terminate_early: False
```

```
10 2.1e+02 ?..!+..!+....!+..!+..?..!+..?..?.....?+....!+.....?...+....?..?..+..?..+.....
2.2 ..?..+....+.....?..+....?..+....+.....?.....?.....+....+..?..+..
2.2 ?..+..?..+....?..+....+.....?..?..+....+.....?..+..?..+....+.....?..
2.2 .....+..?..?..+..+....?..?..+.....?.....
2.2 +....?....?..+....+..?..+....+.....?..+..+..?.....+..?..+..+....+....+...
2.2
```

```
15 Best: 2.1666
total time: 0.012s      mean time: 0.012s
      |                *, 2.17, 2.17, 2.17, 2.17, 2.17
```

```
20 MaxWalkSat
era_length: 50, initialized: 2014-10-06 20:02:44.552281, iterations: 1000, log_eras: True, p_mutation: 0.333333333333, s
earcher: MaxWalkSat, terminate_early: False
```

```
25 2.1e+02 .....++.....+.....
2.0 .....
2.0 .....
2.0 .....
2.0
```

```
30 Best: 2.0079
total time: 0.005s      mean time: 0.005s
      |                *, 2.01, 2.01, 2.01, 2.01, 2.01
```

```
35 =====
=====
```

```
Fonseca
```

```
40 -----
SimulatedAnnealer
era_length: 50, initialized: 2014-10-06 20:02:44.556945, iterations: 1000, log_eras: True, p_mutation: 0.333333333333, s
earcher: SimulatedAnnealer, terminate_early: False
```

```
45 2.0 ?..!+..?..?..+..?..+....!+..?..?.....+..+..?..?..+..?..+.....+....?..+..?..?..+..+..+.....
1.6 !+...?..?..+....+..+..+..?..+..+..+..?..+..+..?..+..?..!+..?..?.....?..+..+..+..+..
1.0 ..?..+..+.....?..+..+..+..+..+..+..+..+..?..+..+..+..?..+..+..+..+..
1.0 ..+..?..+.....+..?..+..+..+..+..+..+..+..?..?.....+.....?..+.....
1.0 ..+.....+.....+..?..+..+..+..+.....+.....?..?.....+.....+.....?..
1.0 +..?.....+..?.....+..?..+..+..?..+..?.....+.....+.....
1.0
```

95 MaxWalkSat
era_length: 50, initialized: 2014-10-06 20:02:44.657892, iterations: 1000, log_eras: True, p_mutation: 0.333333333333, searcher: MaxWalkSat, terminate_early: False

Monday October 06, 2014

hw4/outs.txt

3/13

195

Oct 06, 14 20:03

"csc710sbse: hw4: Witschey"

Page 5/13

```

1.6e+01 .....?+. ....?..+.....?..+..+.....+..?+. ....?....+...
1.6e+01 .+.....?+. ....?..+..+.....+.....?+. ....+.....+.....?....+..+...
200 1.6e+01

```

```

Best: 16.0571
total time: 0.028s      mean time: 0.028s
                    |      *, 16.06, 16.06, 16.06, 16.06, 16.06
205

```

```

MaxWalkSat
era_length: 50, initialized: 2014-10-06 20:02:44.836958, iterations: 1000, log_eras: True, p_mutation: 0.333333333333, s
earcher: MaxWalkSat, terminate_early: False

```

```

210 2.8e+01 +.....
1.7e+01 .....
1.7e+01 .....
1.7e+01 .....
1.7e+01 .....
215

```

```

Best: 17.4286
total time: 0.004s      mean time: 0.004s
                    |      *, 17.43, 17.43, 17.43, 17.43, 17.43
220

```

```

=====
##### Part 5 #####
2014-10-06 20:02:44.841486
225

```

```

Schaffer
-----
SimulatedAnnealer
era_length: 50, initialized: 2014-10-06 20:02:45.359971, iterations: 1000, log_eras: True, p_mutation: 0.333333333333, s
earcher: SimulatedAnnealer, terminate_early: True

```

```

230 2.3e+01 ?....?+. ....?..+.....?..+.....!+. ....?..+.....?....?....?....+...
2.7 .....?..+.....+.....+.....?..+.....+.....?..+...
2.7 ..?..+.....+.....?..+.....!+. ....?..+.....+.....?..+...
2.0 ...+.?..+.....?..+.....?..+.....+.....?..+.....+.....?..+.....+...
235 2.0 ?..+.....+.....?..+.....?..+.....!+. ....?..+.....+.....?..+...
2.0 ?..+.....+.....?..+.....+.....+.....?..+.....+.....?..+...
2.0 ?..+.....+.....+.....?..+.....?..+.....+.....?..+.....+.....?..+...
2.0 .....?..+.....+.....?..+.....+.....+.....?..+.....+.....+...
2.0 ?..+.....+.....+.....?..+.....?..+.....+.....+.....+.....+...
240 2.0

```

```

Best: 2.0504
total time: 0.533s      mean time: 0.018s
                    |      *- , 2.00, 2.00, 2.01, 2.04, 2.17
245

```

MaxWalkSat

```
era_length: 50, initialized: 2014-10-06 20:02:45.490455, iterations: 1000, log_eras: True, p_mutation: 0.333333333333, s
earcher: MaxWalkSat, terminate_early: True

250  2.8e+01 .....
    2.8e+01 .....
    2.8e+01 .....++.....+..
    2.1 .....
    2.1

255  Best:   9.3754
    total time: 0.112s      mean time: 0.004s
    *  ----- |           ,   2.01,   2.22,   4.32,  10.55,  23.34

260  =====
    =====

Fonseca
265  -----
    SimulatedAnnealer
    era_length: 50, initialized: 2014-10-06 20:02:46.314736, iterations: 1000, log_eras: True, p_mutation: 0.333333333333, s
    earcher: SimulatedAnnealer, terminate_early: True

    2.0 !+...!+.?.!+.?......+.+.?.+...+...+...!+.!+.?.?.?...+..?.+..!+.?.+.?....+.?.+.?....
270  1.2 +.?.?.?.+.+.?.?...+...+...+...+.?.?.+...?.?...+..?......+.?.+.+.?.?...
    1.2 !+.?.?.+.?.?...+...+...+...+.?.+.+.?.?...?.+.+.?.?...+..?.+.....
    1.1 ..+.+.+.+.?.+.?.+.+.+.+.?.+.+.+.?.?...+...+...+...+.?.+.?.?...
    1.1 ...+.?.+.+.?.+.?.?...+...+...+...+.?.+.+.+.?.?...+...+...+.?.?...
    1.1 +..?.?...+.?.+.?.+.?.?...+...+...+...+.?.+.+.+.?.?...+...+...+.?.?...
275  1.1

    Best:   1.0431
    total time: 0.828s      mean time: 0.028s
    |           *   ,   1.00,   1.02,   1.03,   1.05,   1.10

280

    MaxWalkSat
    era_length: 50, initialized: 2014-10-06 20:02:46.538240, iterations: 1000, log_eras: True, p_mutation: 0.333333333333, s
    earcher: MaxWalkSat, terminate_early: True

285  2.0 .....+.....
    2.0 .....+.....+...
    1.1 .....
    1.1 .....
    1.1 .....

290  Best:   1.4332
    total time: 0.168s      mean time: 0.006s
    |-- *  ----- ,   1.08,   1.21,   1.33,   1.62,   1.95

295  =====
```

[illegible]

Monday October 06, 2014


```
395 5.7 .....
5.7 .....
5.7 .....
5.7 .....
400 5.7

Best: 5.1809
total time: 0.206s      mean time: 0.007s
----- *| ----- , 2.28, 3.98, 5.01, 6.50, 8.65

405

=====
=====

410 Viennet3
-----
SimulatedAnnealer
era_length: 50, initialized: 2014-10-06 20:02:52.368551, iterations: 1000, log_eras: True, p_mutation: 0.333333333333, s
earcher: SimulatedAnnealer, terminate_early: True

415 2.7e+01 !+...?..?..!+...!+...?..?..!+...?.....?.....?.....+..?..?..+...+..?..+..?..+...+...
1.6e+01 ...?..+..?..?..?..+.....?.....?.....?..+..+..+.....+.....?.....?..?..+..?..+...+..+..
1.6e+01 ?..+...?..?..+...+...?..+...+...+...?..+...+...?..+..?..+...+.....?.....+..!+.....
1.6e+01 ?..?..?..+...+...+...?.....+...+...?..?..+...+.....+...?.....+.....?..+...+...+...
1.6e+01 ..?..?..+..?..+.....+.....?..+...+...+.....?..?..+...+...+...?..+.....+.....
420 1.6e+01 ?.....?..+...+...?..+...+.....+.....+.....+.....+.....?..?..+...+.....+.....+...
1.6e+01 .....?..+...+.....?..+...?.....+.....?..+...+.....?..?..+...+...
1.6e+01 +.....?..+...+...+...+...?..?..?..+...+.....+.....+.....+.....+...?..
1.6e+01

425 Best: 16.0871
total time: 0.967s      mean time: 0.032s
      |                * , 15.98, 16.00, 16.03, 16.07, 16.47

430 MaxWalkSat
era_length: 50, initialized: 2014-10-06 20:02:52.543769, iterations: 1000, log_eras: True, p_mutation: 0.333333333333, s
earcher: MaxWalkSat, terminate_early: True

2.2e+01 .....
2.2e+01 .....+.....
435 1.9e+01 .....+.....++.....
1.7e+01 .....
1.7e+01

Best: 17.7627
440 total time: 0.151s      mean time: 0.005s
      |      -*--- , 16.38, 16.89, 17.30, 17.86, 20.49

=====
```

)	, 3.62,	3.62,	3.71,	3.71,	3.89
)	, 3.83,	3.83,	3.83,	3.83,	3.83
)	, 3.97,	3.97,	3.97,	3.97,	3.97
)	, 4.20,	4.20,	4.20,	4.20,	4.29
)	, 4.21,	4.21,	4.21,	4.21,	4.21
)	, 4.26,	4.26,	4.26,	4.26,	4.26
)	, 4.31,	4.31,	4.31,	4.31,	4.31
)	, 4.31,	4.31,	4.31,	4.31,	4.31
)	, 4.36,	4.36,	4.36,	4.36,	4.36
)	, 4.55,	4.55,	4.55,	4.55,	4.55
)	, 4.58,	4.58,	4.58,	4.58,	4.58
)	, 4.58,	4.58,	4.58,	4.58,	4.58
)	, 4.69,	4.69,	4.69,	4.69,	4.69
)	, 4.69,	4.69,	4.69,	4.69,	4.69
)	, 4.74,	4.74,	4.74,	4.74,	4.74
)	, 4.75,	4.75,	4.75,	4.75,	4.75
)	, 4.79,	4.79,	4.79,	4.85,	4.92
)	, 4.97,	4.97,	4.97,	4.97,	4.97
)	, 5.06,	5.06,	5.06,	5.06,	5.06
)	, 5.09,	5.09,	5.09,	5.09,	5.09
)	, 5.21,	5.21,	5.21,	5.21,	5.21
)	, 5.37,	5.37,	5.37,	5.37,	5.37
)	, 5.40,	5.40,	5.40,	5.40,	5.40
)	, 5.62,	5.62,	5.62,	5.62,	5.62
)	, 5.70,	5.70,	5.70,	5.70,	5.70
)	, 5.94,	5.94,	5.94,	5.94,	5.94
)	, 5.94,	5.94,	5.94,	5.94,	5.94
)	, 6.38,	6.38,	6.38,	6.38,	6.38
)	, 6.39,	6.39,	6.39,	6.39,	6.39
)	, 6.42,	6.42,	6.42,	6.42,	6.42
)	, 6.44,	6.44,	6.44,	6.44,	6.44
)	, 6.45,	6.45,	6.45,	6.45,	6.45
)	, 6.52,	6.52,	6.52,	6.52,	6.52
)	, 6.62,	6.62,	6.62,	6.62,	6.62
)	, 6.66,	6.66,	6.66,	6.66,	6.66
)	, 6.66,	6.66,	6.66,	6.66,	6.66
)	, 6.67,	6.67,	6.67,	6.67,	6.67
)	, 6.67,	6.67,	6.67,	6.67,	6.67
)	, 6.69,	6.69,	6.69,	6.69,	6.69
)	, 6.71,	6.71,	6.71,	6.71,	6.71
)	, 6.72,	6.72,	6.72,	6.72,	6.72
)	, 6.76,	6.76,	6.76,	6.76,	6.76
)	, 6.82,	6.82,	6.82,	6.82,	6.82
)	, 6.86,	6.86,	6.86,	6.86,	6.86
)	, 6.90,	6.90,	6.90,	6.90,	6.90
)	, 6.92,	6.92,	6.92,	6.92,	6.92
)	, 6.93,	6.93,	6.93,	6.93,	6.93

Oct 06, 14 20:03

"csc710sbse: hw4: Witschey"

Page 11/13

500	24	,	SA	,	6	,	0	(*)	,	6.96	,	6.96	,	6.96	,	6.96	,	6.96
	24	,	SA	,	6	,	0	(*)	,	6.96	,	6.96	,	6.96	,	6.96	,	6.96
	25	,	SA	,	7	,	0	(*)	,	7.04	,	7.04	,	7.04	,	7.04	,	7.04
	25	,	SA	,	7	,	0	(*)	,	7.06	,	7.06	,	7.06	,	7.06	,	7.06
	25	,	SA	,	7	,	0	(*)	,	7.06	,	7.06	,	7.06	,	7.06	,	7.06
	25	,	SA	,	7	,	0	(*)	,	7.06	,	7.06	,	7.06	,	7.06	,	7.06
	25	,	SA	,	7	,	0	(*)	,	7.06	,	7.06	,	7.06	,	7.06	,	7.06
	25	,	SA	,	7	,	0	(*)	,	7.07	,	7.07	,	7.07	,	7.07	,	7.07
505	26	,	SA	,	7	,	0	(*)	,	7.09	,	7.09	,	7.09	,	7.09	,	7.09
	26	,	SA	,	7	,	0	(*)	,	7.10	,	7.10	,	7.10	,	7.10	,	7.10
	26	,	SA	,	7	,	0	(*)	,	7.10	,	7.10	,	7.10	,	7.10	,	7.10
	26	,	SA	,	7	,	0	(*)	,	7.10	,	7.10	,	7.10	,	7.10	,	7.10
	26	,	SA	,	7	,	0	(*)	,	7.11	,	7.11	,	7.11	,	7.11	,	7.11
510	26	,	SA	,	7	,	0	(*)	,	7.11	,	7.11	,	7.11	,	7.11	,	7.11
	27	,	SA	,	7	,	0	(*)	,	7.12	,	7.12	,	7.12	,	7.12	,	7.12
	27	,	SA	,	7	,	0	(*)	,	7.13	,	7.13	,	7.13	,	7.13	,	7.13
	27	,	SA	,	7	,	0	(*)	,	7.13	,	7.13	,	7.13	,	7.13	,	7.13
	27	,	SA	,	7	,	0	(*)	,	7.14	,	7.14	,	7.14	,	7.14	,	7.14
515	28	,	SA	,	7	,	0	(*)	,	7.16	,	7.16	,	7.16	,	7.16	,	7.16
	28	,	SA	,	7	,	0	(*)	,	7.18	,	7.18	,	7.18	,	7.18	,	7.18
	28	,	SA	,	7	,	0	(*)	,	7.18	,	7.18	,	7.18	,	7.18	,	7.18
	28	,	SA	,	7	,	0	(*)	,	7.19	,	7.19	,	7.19	,	7.19	,	7.19
	28	,	SA	,	7	,	0	(*)	,	7.20	,	7.20	,	7.20	,	7.20	,	7.20
520	29	,	SA	,	7	,	0	(*)	,	7.24	,	7.24	,	7.24	,	7.24	,	7.24
	29	,	SA	,	7	,	0	(*)	,	7.25	,	7.25	,	7.25	,	7.25	,	7.25
	29	,	SA	,	7	,	0	(*)	,	7.26	,	7.26	,	7.26	,	7.26	,	7.26
	29	,	SA	,	7	,	0	(*)	,	7.26	,	7.26	,	7.26	,	7.26	,	7.26
	30	,	SA	,	7	,	0	(*)	,	7.28	,	7.28	,	7.28	,	7.28	,	7.28
525	30	,	SA	,	7	,	0	(*)	,	7.28	,	7.28	,	7.28	,	7.28	,	7.28
	31	,	SA	,	7	,	0	(*)	,	7.31	,	7.31	,	7.31	,	7.31	,	7.31
	31	,	SA	,	7	,	0	(*)	,	7.31	,	7.31	,	7.31	,	7.31	,	7.31
	31	,	SA	,	7	,	0	(*)	,	7.31	,	7.31	,	7.31	,	7.31	,	7.31
	31	,	SA	,	7	,	0	(*)	,	7.34	,	7.34	,	7.34	,	7.34	,	7.34
530	32	,	SA	,	7	,	0	(*)	,	7.37	,	7.37	,	7.37	,	7.37	,	7.37
	32	,	SA	,	7	,	0	(*)	,	7.38	,	7.38	,	7.38	,	7.38	,	7.38
	33	,	SA	,	7	,	0	(*)	,	7.44	,	7.44	,	7.44	,	7.44	,	7.44
	33	,	SA	,	7	,	0	(*)	,	7.45	,	7.45	,	7.45	,	7.45	,	7.45
	34	,	SA	,	7	,	0	(*)	,	7.51	,	7.51	,	7.51	,	7.51	,	7.51
535	34	,	SA	,	7	,	0	(*)	,	7.56	,	7.56	,	7.56	,	7.56	,	7.56
	34	,	SA	,	7	,	0	(*)	,	7.57	,	7.57	,	7.57	,	7.57	,	7.57
	35	,	SA	,	7	,	0	(*)	,	7.76	,	7.76	,	7.76	,	7.76	,	7.76
	35	,	SA	,	7	,	0	(*)	,	7.77	,	7.77	,	7.77	,	7.77	,	7.77
	35	,	MWS	,	7	,	0	(*)	,	7.92	,	7.92	,	7.92	,	7.92	,	7.92
540	36	,	MWS	,	8	,	0	(*)	,	8.15	,	8.15	,	8.15	,	8.15	,	8.15
	36	,	MWS	,	8	,	0	(*)	,	8.16	,	8.16	,	8.16	,	8.16	,	8.16
	36	,	MWS	,	8	,	0	(*)	,	8.19	,	8.19	,	8.19	,	8.19	,	8.19
	36	,	MWS	,	8	,	0	(*)	,	8.20	,	8.20	,	8.20	,	8.20	,	8.20
	36	,	MWS	,	8	,	0	(*)	,	8.21	,	8.21	,	8.21	,	8.21	,	8.21
545	36	,	MWS	,	8	,	0	(*)	,	8.31	,	8.31	,	8.31	,	8.31	,	8.31
	37	,	MWS	,	8	,	0	(*)	,	8.41	,	8.41	,	8.41	,	8.41	,	8.41
	37	,	MWS	,	8	,	0	(*)	,	8.42	,	8.42	,	8.42	,	8.42	,	8.42
	37	,	SA	,	8	,	0	(*)	,	8.43	,	8.43	,	8.43	,	8.43	,	8.43

Oct 06, 14 20:03		"csc710sbse: hw4: Witschey"					Page 12/13				
550	37 ,	MWS ,	8 ,	0 (*) ,	8.45 ,	8.45 ,	8.45 ,	8.45 ,	8.45
	38 ,	SA ,	8 ,	0 (*) ,	8.47 ,	8.47 ,	8.47 ,	8.47 ,	8.47
	38 ,	MWS ,	8 ,	0 (*) ,	8.48 ,	8.48 ,	8.48 ,	8.48 ,	8.48
	39 ,	MWS ,	8 ,	0 (*) ,	8.49 ,	8.49 ,	8.49 ,	8.49 ,	8.49
	39 ,	MWS ,	8 ,	0 (*) ,	8.50 ,	8.50 ,	8.50 ,	8.50 ,	8.50
555	40 ,	MWS ,	8 ,	0 (*) ,	8.51 ,	8.51 ,	8.51 ,	8.51 ,	8.51
	40 ,	MWS ,	8 ,	0 (*) ,	8.53 ,	8.53 ,	8.53 ,	8.53 ,	8.53
	41 ,	MWS ,	8 ,	0 (*) ,	8.58 ,	8.58 ,	8.58 ,	8.58 ,	8.58
	41 ,	MWS ,	8 ,	0 (*) ,	8.58 ,	8.58 ,	8.58 ,	8.58 ,	8.58
	41 ,	MWS ,	8 ,	0 (*) ,	8.59 ,	8.59 ,	8.59 ,	8.59 ,	8.59
560	41 ,	MWS ,	8 ,	0 (*) ,	8.59 ,	8.59 ,	8.59 ,	8.59 ,	8.59
	41 ,	MWS ,	8 ,	0 (*) ,	8.60 ,	8.60 ,	8.60 ,	8.60 ,	8.60
	41 ,	MWS ,	8 ,	0 (*) ,	8.62 ,	8.62 ,	8.62 ,	8.62 ,	8.62
	42 ,	MWS ,	8 ,	0 (*) ,	8.67 ,	8.67 ,	8.67 ,	8.67 ,	8.67
	42 ,	MWS ,	8 ,	0 (*) ,	8.68 ,	8.68 ,	8.68 ,	8.68 ,	8.68
565	43 ,	MWS ,	8 ,	0 (*) ,	8.73 ,	8.73 ,	8.73 ,	8.73 ,	8.73
	43 ,	MWS ,	8 ,	0 (*) ,	8.74 ,	8.74 ,	8.74 ,	8.74 ,	8.74
	43 ,	MWS ,	8 ,	0 (*) ,	8.81 ,	8.81 ,	8.81 ,	8.81 ,	8.81
	44 ,	MWS ,	8 ,	0 (*) ,	8.91 ,	8.91 ,	8.91 ,	8.91 ,	8.91
	44 ,	MWS ,	8 ,	0 (*) ,	8.92 ,	8.92 ,	8.92 ,	8.92 ,	8.92
570	45 ,	MWS ,	8 ,	0 (*) ,	8.94 ,	8.94 ,	8.94 ,	8.94 ,	8.94
	45 ,	MWS ,	8 ,	0 (*) ,	8.95 ,	8.95 ,	8.95 ,	8.95 ,	8.95
	46 ,	MWS ,	9 ,	0 (*) ,	9.04 ,	9.04 ,	9.04 ,	9.04 ,	9.04
	46 ,	MWS ,	9 ,	0 (*) ,	9.05 ,	9.05 ,	9.05 ,	9.05 ,	9.05
	46 ,	MWS ,	9 ,	0 (*) ,	9.07 ,	9.07 ,	9.07 ,	9.07 ,	9.07
575	47 ,	MWS ,	9 ,	0 (*) ,	9.10 ,	9.10 ,	9.10 ,	9.10 ,	9.10
	47 ,	MWS ,	9 ,	0 (*) ,	9.12 ,	9.12 ,	9.12 ,	9.12 ,	9.12
	48 ,	MWS ,	9 ,	0 (*) ,	9.20 ,	9.20 ,	9.20 ,	9.20 ,	9.20
	48 ,	MWS ,	9 ,	0 (*) ,	9.21 ,	9.21 ,	9.21 ,	9.21 ,	9.21
	48 ,	MWS ,	9 ,	0 (*) ,	9.21 ,	9.21 ,	9.21 ,	9.21 ,	9.21
580	49 ,	MWS ,	9 ,	0 (*) ,	9.25 ,	9.25 ,	9.25 ,	9.25 ,	9.25
	49 ,	MWS ,	9 ,	0 (*) ,	9.26 ,	9.26 ,	9.26 ,	9.26 ,	9.26
	49 ,	MWS ,	9 ,	0 (*) ,	9.29 ,	9.29 ,	9.29 ,	9.29 ,	9.29
	50 ,	MWS ,	9 ,	0 (*) ,	9.37 ,	9.37 ,	9.37 ,	9.37 ,	9.37
	50 ,	MWS ,	9 ,	0 (*) ,	9.37 ,	9.37 ,	9.37 ,	9.37 ,	9.37
585	51 ,	MWS ,	9 ,	0 (*) ,	9.38 ,	9.38 ,	9.38 ,	9.38 ,	9.38
	51 ,	MWS ,	9 ,	0 (*) ,	9.39 ,	9.39 ,	9.39 ,	9.39 ,	9.39
	52 ,	MWS ,	9 ,	0 (*) ,	9.42 ,	9.42 ,	9.42 ,	9.42 ,	9.42
	52 ,	MWS ,	9 ,	0 (*) ,	9.43 ,	9.43 ,	9.43 ,	9.43 ,	9.43
	52 ,	MWS ,	9 ,	0 (*) ,	9.43 ,	9.43 ,	9.43 ,	9.43 ,	9.43
590	52 ,	MWS ,	9 ,	0 (*) ,	9.45 ,	9.45 ,	9.45 ,	9.45 ,	9.45
	53 ,	MWS ,	9 ,	0 (*) ,	9.48 ,	9.48 ,	9.48 ,	9.48 ,	9.48
	53 ,	MWS ,	9 ,	0 (*) ,	9.51 ,	9.51 ,	9.51 ,	9.51 ,	9.51
	53 ,	MWS ,	9 ,	0 (*) ,	9.54 ,	9.54 ,	9.54 ,	9.54 ,	9.54
	54 ,	MWS ,	9 ,	0 (*) ,	9.67 ,	9.67 ,	9.67 ,	9.67 ,	9.67
595	54 ,	MWS ,	9 ,	0 (*) ,	9.72 ,	9.72 ,	9.72 ,	9.72 ,	9.72
	54 ,	MWS ,	9 ,	0 (*) ,	9.83 ,	9.83 ,	9.83 ,	9.83 ,	9.83
	55 ,	MWS ,	9 ,	0 (*) ,	9.89 ,	9.89 ,	9.89 ,	9.89 ,	9.89
	55 ,	MWS ,	9 ,	0 (*) ,	9.91 ,	9.91 ,	9.91 ,	9.91 ,	9.91
	55 ,	MWS ,	9 ,	0 (*) ,	9.94 ,	9.94 ,	9.94 ,	9.94 ,	9.94
600	56 ,	MWS ,	9 ,	0 (*) ,	9.97 ,	9.97 ,	9.97 ,	9.97 ,	9.97
	56 ,	MWS ,	10 ,	0 (*) ,	10.00 ,	10.00 ,	10.00 ,	10.00 ,	10.00

Oct 06, 14 20:03

"csc710sbse: hw4: Witschey"

Page 13/13

57	,	MWS	,	10	,	0	(*)	,10.02,	10.02,	10.02,	10.02,	10.02
57	,	MWS	,	10	,	0	(*)	,10.05,	10.05,	10.05,	10.05,	10.05
58	,	MWS	,	10	,	0	(*)	,10.10,	10.10,	10.10,	10.10,	10.10
58	,	MWS	,	10	,	0	(*)	,10.16,	10.16,	10.16,	10.16,	10.16
605	58	,	MWS	,	10	,	0	*)	,10.17,	10.17,	10.17,	10.17,	10.17
	59	,	MWS	,	10	,	0	*)	,10.23,	10.23,	10.23,	10.23,	10.23
	59	,	MWS	,	10	,	0	*)	,10.25,	10.25,	10.25,	10.25,	10.25
	59	,	MWS	,	10	,	0	*)	,10.25,	10.25,	10.25,	10.25,	10.25
	60	,	MWS	,	10	,	0	*)	,10.33,	10.33,	10.33,	10.33,	10.33
610	60	,	MWS	,	10	,	0	*)	,10.35,	10.35,	10.35,	10.35,	10.35
	60	,	MWS	,	10	,	0	*)	,10.35,	10.35,	10.35,	10.35,	10.35
	60	,	MWS	,	10	,	0	*)	,10.36,	10.36,	10.36,	10.36,	10.36
	60	,	MWS	,	10	,	0	*)	,10.42,	10.42,	10.42,	10.42,	10.42
	61	,	MWS	,	10	,	0	*)	,10.54,	10.54,	10.54,	10.54,	10.54
615	61	,	MWS	,	10	,	0	*)	,10.55,	10.55,	10.55,	10.55,	10.55
	62	,	MWS	,	10	,	0	*)	,10.58,	10.58,	10.58,	10.58,	10.58
	62	,	MWS	,	10	,	0	*)	,10.59,	10.59,	10.59,	10.59,	10.59
	62	,	MWS	,	10	,	0	*)	,10.60,	10.60,	10.60,	10.60,	10.60
	63	,	MWS	,	10	,	0	*)	,10.67,	10.67,	10.67,	10.67,	10.67
620	63	,	MWS	,	10	,	0	*)	,10.69,	10.69,	10.69,	10.69,	10.69
	64	,	MWS	,	10	,	0	*)	,10.77,	10.77,	10.77,	10.77,	10.77
	64	,	MWS	,	10	,	0	*)	,10.77,	10.77,	10.77,	10.77,	10.77
	64	,	MWS	,	10	,	0	*)	,10.78,	10.78,	10.78,	10.78,	10.78
	64	,	MWS	,	10	,	0	*)	,10.89,	10.89,	10.89,	10.89,	10.89
625	64	,	MWS	,	10	,	0	*)	,10.99,	10.99,	10.99,	10.99,	10.99
	65	,	MWS	,	11	,	0	*)	,11.15,	11.15,	11.15,	11.15,	11.15
	65	,	MWS	,	11	,	0	*)	,11.30,	11.30,	11.30,	11.30,	11.30
	65	,	MWS	,	11	,	0	*)	,11.36,	11.36,	11.36,	11.36,	11.36
630	65	,	MWS	,	11	,	0	*)	,11.96,	11.96,	11.96,	11.96,	11.96
	None													

Oct 06, 14 16:53 "csc710sbse: hw4: Witschey" Page 1/6

```

"""## Log Stuff

Adapted from [Dr. Tim Menzies' logging code](https://github.com/timm/sbse14/blob/master/log.py).

5 Logs are places to store records of past events. There are two types of logs:

+ _Num_ : for numbers
+ _Sym_ : for everything else.

10 Those logs can be queried to find e.g. the highest
and lowest value of the number seen so far. Alternatively,
they can be queried to return values at the same probability
as the current log contents.

15 ### Max Log Size

To avoid logs consuming all memory, logs store at
most _The.cache.keep_ entries (e.g. 128):

20 + If more
than that number of entries arrive, then some old
entry (selected at random) will be deleted.
+ The nature of this cache means that some rare
events might be missed. To check for that, running
the code multiple times and, each time, double the
25 cache size. Stop when doubling the cache size stops
changing the output.

Just as an example of that process, here we are logging 1,000,000 numbers in a l
og with a cache of size 16.
30 Note that the resulting cache is much smaller than 1,000,000 items. Also, the co
ntents of the cache
come from the entire range one to one million (so our log is not biased to just
the first few samples:

% python -i log.py
>>> The.cache.keep = 16
35 >>> log = Num()
>>> for x in xrange(1000000): log += x
>>> sorted(log._cache)
[77748, 114712, 122521, 224268,
289880, 313675, 502464, 625036,
40 661881, 663207, 680085, 684674,
867075, 875594, 922141, 945896]
>>>

### Caching Slow Reports

45 Some of the things we want to report from these logs take a little while to cal
culate (e.g. finding the median
requires a sort of a numeric cache):

+ Such reports should be run and cached so they can be accessed many time without
the need
50 for tedious recalculation.
+ These reports become outdated if new log information arrives so the following
code deletes these reports if ever new data arrives.
+ The protocol for access those reports is to call _log.has().x_ where "x" is a
field
generated by the report. Log subclasses generate reports using the special _rep
ort()_ method
55 (see examples, below).

Just as an example of reporting, after the above run (where we logged 1,000,000
numbers), the following reports are available:

>>> log.has().lo
60 0
>>> log.has().hi
945896
>>> print log.has().median # 50th percentile
662544.0

```

Oct 06, 14 16:53 "csc710sbse: hw4: Witschey" Page 2/6

```

65 >>> print log.has().iqr # (75-25)th percentile
205194

Note that our median is not as expected (it should be around half a million). Wh
y? Well, clearly a cache of size 16 is
too small to track a million numbers. So how many numbers do we need? Well, that
depends on the distribution being explored
70 but here's how the median is effected by cache size for uniform distributions:

>>> for size in [16,32,64,128,256]:
...     The.cache.keep=size
...     log = Num()
75 ...     for x in xrange(1000000): log += x
...     print size, ":" log.has().median
...
16 : 637374.5
32 : 480145.5
80 64 : 520585.5
128 : 490742.0
256 : 470870.5

85 Note that we get pretty close to half a million with cache sizes at 32 or above.
And the lesson: sometimes, a limited
sample can offer a useful approximation to a seemingly complex process.

## Standard Header
"""
90 from __future__ import division, print_function
import sys, random, math, datetime, time, re
from base import memo
import base
import functools

95 class Log(object):
    "Keep a random sample of stuff seen so far."

    def __init__(self, inits=None, label=None, max_size=256):
100     self._cache = []
        self._n = 0
        self._report = None
        self.label = label or ''
        self.max_size = max_size
105     self._valid_statistics = False
        if inits:
            map(self._iadd_, inits)

    def random_index(self):
110     return base.random_index(self._cache)

    def _iadd_(self, x):
        if x is None:
            return x
115
        if isinstance(x, Log):
            map(self._iadd_, x._cache)

        self._n += 1
        changed = False

        # if cache has room, add item
        if self.max_size is None or len(self._cache) < self.max_size:
            changed = True
            self._cache.append(x)
125
        # cache is full: maybe replace an old item
        else:
            # items less likely to be replaced later in the run:
            # leads to uniform sample of entire run
            if random.random() <= self.max_size / self._n:
                changed = True
                self._cache[self.random_index()] = x
130

        if changed:

```

Oct 06, 14 16:53

"csc710sbse: hw4: Witschey"

Page 3/6

```

135         self._invalidate_statistics()
            self._change(x)

            return self

140     def __add__(self, x):
        inits = self._cache + x._cache
        return NumberLog(inits=inits, label='generated via __add__', max_size=No
ne)

145     def any(self):
        return random.choice(self._cache)

150     def report(self):
        if self._report is None:
            self._report = self.generate_report()
        return self._report

155     def setup(self):
        raise NotImplementedError()

160     def contents(self):
        # slow, but most generic copy implementation
        return copy.deepcopy(self._cache)

165     def _invalidate_statistics(self):
        '''
        default implementation. if _valid_statistics is something other than
        a boolean, reimplement!
        '''
        self._valid_statistics = False

170     def ish(self, *args, **kwargs):
        raise NotImplementedError()

175     def _change(self, x):
        '''
        override to add incremental updating functionality
        '''
        pass

180     def _prepare_data(self):
        s = '_prepare_data()' not implemented for ' + self.__class__.__name__
        raise NotImplementedError(s)

185     @staticmethod
        def log_for(t):
            if t == int or t == float or isinstance(t, (int, float)):
                return NumberLog()
            else:
                return SymbolLog()

190     def statistic(f):
        '''
        decorator for log functions that return statistics about contents.
        if _valid_statistics is False, generate valid stats before calling
        the wrapped function.
        '''
        @functools.wraps(f)
        def wrapper(*args, **kwargs):
            self = args[0]
            if not self._valid_statistics:
                self._prepare_data()
            return f(*args, **kwargs)

200     return wrapper

'''
### Num
A _Num_ is a _Log_ for numbers.
'''

```

Oct 06, 14 16:53

"csc710sbse: hw4: Witschey"

Page 4/6

```

+ Tracks _lo_ and _hi_ values.
+ Reports median and the IQR the (75-25)th range.
210 + Generates numbers from the log by a three-way interpolation (see _ish()).

'''
class NumberLog(Log):
215     def __init__(self, *args, **kwargs):
        super(NumberLog, self).__init__(*args, **kwargs)
        assert self._n == 0

220         # set to values that will be immediately overridden
        self.lo, self.hi = sys.maxint, -sys.maxint

225     def _change(self, x):
        # update lo,hi
        self.lo = min(self.lo, x)
        self.hi = max(self.hi, x)

230     def _prepare_data(self):
        if not self._valid_statistics:
            self._cache.sort()
            self._valid_statistics = True

235     def contents(self):
        return list(self._cache)

240     def norm(self,x):
        "normalize the argument with respect to maximum and minimum"
        if self.hi == self.lo:
            raise ValueError('hi and lo of {} are equal'.format(self.__name__))
        return (x - self.lo) / (self.hi - self.lo)

245     def generate_report(self):
        return memo(median=self.median(), iqr=self.iqr(),
                    lo=self.lo, hi=self.hi)

250     def ish(self,f=0.1):
        """return a num likely to be similar to/representative of
        nums in the distribution"""
        return self.any() + f*(self.any() - self.any())

255     @statistic
        def median(self):
            # implementation from http://stackoverflow.com/a/10482734/3408454
            n = len(self._cache)

            if n % 2:
                return self._cache[n // 2]

            return (self._cache[n // 2] + self._cache[n // 2 - 1]) / 2

260     def mean(self):
        n = len(self._cache)
        return sum(self._cache) / n

265     @statistic
        def iqr(self):
            n = len(self._cache)
            return self._cache[int(n*.75)] - self._cache[int(n*.5)]

270     def total(self):
        return sum(self._cache)

275     def better(self, log2):
        if not self._cache or not log2._cache: return False
        if self.median() < log2.median(): return True
        if self.iqr() < log2.iqr(): return True
        return False

```

Oct 06, 14 16:53

"csc710sbse: hw4: Witschey"

Page 5/6

```

280     @statistic
    def xtile(self, lo=0, hi=0.001,
              width=50,
              chops=[0.1, 0.3, 0.5, 0.7, 0.9],
285             marks=["-", " ", " ", " ", "-"],
              bar="|", star="*",
              show="{: >6.2f}"):
        """The function _xtile_ takes a list of (possibly)
        unsorted numbers and presents them as a horizontal
        xtile chart (in ascii format). The default is a
        contracted _quintile_ that shows the
        10,30,50,70,90 breaks in the data (but this can be
        changed- see the optional flags of the function).
        """

295         lo = min(lo, self._cache[0])
        hi = max(hi, self._cache[-1])
        if hi == lo:
            hi = hi + .001 # ugh

300         pos = lambda p: self._cache[int(len(self._cache) * p)]
        place = lambda x: min(width-1, int(width * float((x - lo))/(hi - lo)))
        pretty = lambda xs: ' '.join([show.format(x) for x in xs])

305         what = [pos(p) for p in chops]
        where = [place(n) for n in what]

        out = [' ' * width

310         for one,two in base.pairs(where):
            for i in range(one, two):
                out[i] = marks[0]
            marks = marks[1:]

315         out[int(width / 2)] = bar
        out[place(pos(0.5))] = star

        return ''.join(out) + ", " + pretty(what)

320     """

    WARNING: the call to _sorted_ in _report() makes this code
    a candidate for a massive CPU suck (it is always sorting newly arrived data).
325    So distinguish between _adding_ things to a log in the _last_ era and
    using that information in the _next_ era (so the log from the last era
    is staple in the current).

    ### Sym

330    A _Sym_ is a _Log_ for non-numerics.

    + Tracks frequency counts for symbols, and the most common symbol (the _mode_);
    + Reports the entropy of the space (a measure of diversity: lower values mean fe
    wer rarer symbols);
335    + Generated symbols from the log by returning symbols at the same probability of
    the frequency counts (see _ish()).

    """
    class SymbolLog(Log):

340        @property
        def valid_statistics(self):
            return self._counts is None

        def _invalidate_statistics(self):
345            # '_counts is None' => invalidation of calculated statistics
            # _mode would be a bad idea: what's the 'null' equivalent,
            # when None is a valid index into _counts?
            self._counts = None

350        def _prepare_data(self):

```

Oct 06, 14 16:53

"csc710sbse: hw4: Witschey"

Page 6/6

```

        counts = {}
        mode = None
        mode_count = 0

355        for x in self._cache:
            c = counts[x] = counts.get(x, 0) + 1
            if c > mode_count:
                mode = x

360        self._counts, self._mode = counts, mode
        return self._counts, self._mode

    @statistic
    def counts(self):
365        return self._counts

    @statistic
    def mode(self):
        return self._mode

370    @statistic
    def distribution(self):
        return {k: v / len(self._cache) for k, v in self.counts().items()}

375    def generate_report(self):
        return memo(
            distribution = self.distribution(),
            entropy = self.entropy(),
            mode = self.mode())

380    @statistic
    def ish(self):
        tmp = 0
        threshold = random.random()
385        for k, v in self.distribution().items():
            tmp += v
            if tmp >= threshold:
                return k

        # this shouldn't happen, but just in case...
390        return random.choice(self._cache)

    @statistic
    def entropy(self,e=0):
        n = len(self._cache)
395        for k, v in self.counts().items():
            p = v / n
            # TODO: understand this equation better
            e -= p * math.log(p, 2) if p else 0
        return e

```


Oct 06, 14 15:33

"csc710sbse: hw4: Witschey"

Page 1/1

```

from independent_variable import IndependentVariable; del independent_variable
from schaffer import Schaffer; del schaffer
from kursawe import Kursawe; del kursawe
from fonseca import Fonseca; del fonseca
5 from zdt1 import ZDT1; del zdt1
  from zdt3 import ZDT3; del zdt3
  from viennet3 import Viennet3; del viennet3

del model

```

Sep 30, 14 12:27

"csc710sbse: hw4: Witschey"

Page 1/1

```

# all adapted from Dr. Tim Menzies' model code:
# https://github.com/timm/sbse14/blob/master/models.py

class Model(object):
5   def __init__(self, independents=None, dependents=None,
      energy_min=None, energy_max=None, enforce_energy_constraints=False):
      if independents is None or dependents is None:
          raise ValueError

10      self.xs = independents
      self.ys = dependents
      self.energy_max = energy_max
      self.energy_min = energy_min
      self.enforce_energy_constraints = enforce_energy_constraints

15   def normalize(self, x):
      n = x - self.energy_min
      d = self.energy_max - self.energy_min
      try:
          return n / d
20      except ZeroDivisionError:
          return 0.5

   def random_input_vector(self):
25       return tuple(x() for x in self.xs)

   def __call__(self, v, norm=False):
      energy_vector = tuple(y(v) for y in self.ys)
      energy_total = sum(energy_vector)

30      if self.enforce_energy_constraints:
          energy_errmsg = 'current energy {} not in range [{}, {}]' .format(
              energy_total, self.energy_min, self.energy_max)

35      if self.energy_min is None or self.energy_min > energy_total:
          if self.enforce_energy_constraints:
              raise ValueError(energy_errmsg)
          self.energy_min = energy_total

40      if self.energy_max is None or energy_total > self.energy_max:
          if self.enforce_energy_constraints:
              raise ValueError(energy_errmsg)
          self.energy_max = energy_total

45      return energy_vector

   def energy(self, energy_vector):
      return sum(energy_vector)

```

Sep 17, 14 21:54

"csc710sbse: hw4: Witschey"

Page 1/1

```

# all adapted from Dr. Tim Menzies' model code:
# https://github.com/timm/sbse14/blob/master/models.py

import random

5 class IndependentVariable(object):
    def __init__(self, min=None, max=None, type=float):
        self.min = min
        self.max = max
10        self.type = type

    def __call__(self):
        if self.type == float:
            f = random.uniform
15        elif self.type == int:
            f = random.randint

        return f(self.min, self.max)

```

Sep 22, 14 23:27

"csc710sbse: hw4: Witschey"

Page 1/1

```

# all adapted from Dr. Tim Menzies' model code:
# https://github.com/timm/sbse14/blob/master/models.py

from __future__ import division
5 import math

from model import Model
from independent_variable import IndependentVariable as IV
from witschey.base import memo_sqrt

10 class Fonseca(Model):
    def __init__(self, ivs=3):
        ivs = tuple(IV(min=-4, max=4) for _ in xrange(ivs - 1))

15    def f1(xs):
        e = sum((x - (1 / memo_sqrt(i+1))) ** 2 for i, x in enumerate(xs))
        return 1 - math.exp(-e)

    def f2(xs):
20        e = sum((x + (1 / memo_sqrt(i+1))) ** 2 for i, x in enumerate(xs))
        return 1 - math.exp(-e)

    super(Fonseca, self).__init__(independents=ivs, dependents=(f1, f2))

```

Sep 22, 14 23:13

"csc710sbse: hw4: Witschey"

Page 1/1

```

# all adapted from Dr. Tim Menzies' model code:
# https://github.com/timm/sbse14/blob/master/models.py

from __future__ import division
5 import math

from model import Model
from independent_variable import IndependentVariable as IV

10 class Kursawe(Model):
    def __init__(self, ivs=3, a=0.8, b=3):
        ivs = tuple(IV(min=-5, max=5) for _ in xrange(ivs - 1))
        self.a = a
        self.b = b

15     def f1(xs):
        rv = 0
        for i in xrange(len(xs) - 1):
            exponent = (-0.2) * math.sqrt(xs[i] ** 2 + xs[i+1] ** 2)
20             rv += -10 * math.exp(exponent)
        return rv

    def f2(xs):
        f = lambda x: (math.fabs(x)**self.a) + (5 * math.sin(x)**self.b)
25         return sum(f(x) for x in xs)

    super(Kursawe, self).__init__(independents=ivs, dependents=(f1, f2))

```

Sep 22, 14 23:09

"csc710sbse: hw4: Witschey"

Page 1/1

```

# all adapted from Dr. Tim Menzies' model code:
# https://github.com/timm/sbse14/blob/master/models.py

from model import Model
5 from independent_variable import IndependentVariable as IV

class Schaffer(Model):
    def __init__(self, ivs=1):
        ivs = tuple(IV(min=-10^5, max=10^5) for _ in xrange(ivs))
10         # we use def instead of lambdas so the functions keep their __name__s
        def f1(xs):
            return sum(x ** 2 for x in xs)
        def f2(xs):
            return sum((x - 2) ** 2 for x in xs)

15         super(Schaffer, self).__init__(
            independents=ivs, dependents=(f1, f2))

```

Sep 22, 14 23:13

"csc710sbse: hw4: Witschey"

Page 1/1

```

# all adapted from Dr. Tim Menzies' model code:
# https://github.com/timm/sbse14/blob/master/models.py

from __future__ import division
5 import math

from model import Model
from independent_variable import IndependentVariable as IV

10 class Viennet3(Model):
    def __init__(self):

        def f1(xs):
            x_1sq = xs[0] ** 2
            x_2sq = xs[1] ** 2
            a = 0.5 * x_1sq
            b = math.sin(x_1sq + x_2sq)
            return a + x_2sq + b

20        def f2(xs):
            x_1 = xs[0]
            x_2 = xs[1]

            a = ((3 * x_1 - 2 * x_2 + 4) ** 2) / 8
            b = ((x_1 + x_2 + 1) ** 2) / 27

            return a + b + 15

30        def f3(xs):
            x_1sq = xs[0] ** 2
            x_2sq = xs[1] ** 2

            a = 1 / (x_1sq + x_2sq + 1)
            b = 1.1 * math.exp(-x_1sq - x_2sq)

            return a - b

40        ivs = (IV(min=-3, max=3), IV(min=-3, max=3))
        super(Viennet3, self).__init__(
            independents=ivs, dependents=(f1, f2, f3))

```

Sep 22, 14 23:13

"csc710sbse: hw4: Witschey"

Page 1/1

```

# all adapted from Dr. Tim Menzies' model code:
# https://github.com/timm/sbse14/blob/master/models.py

from __future__ import division
5 import math

from model import Model
from independent_variable import IndependentVariable as IV

10 class ZDT1(Model):
    def __init__(self, ivs=30):

        def g(xs):
            return 1 + 9 * sum(xs[1:]) / (len(xs) - 1)

15        def f1(xs):
            return xs[0]

        def f2(xs):
            gxs = g(xs)
            return gxs * (1 - math.sqrt(xs[0] / gxs))

20        ivs = tuple(IV(min=0, max=1) for _ in xrange(30))
        super(ZDT1, self).__init__(independents=ivs, dependents=(f1, f2, g))

```

Sep 22, 14 23:12

"csc710sbse: hw4: Witschey"

Page 1/1

```

# all adapted from Dr. Tim Menzies' model code:
# https://github.com/timm/sbse14/blob/master/models.py

from __future__ import division
5   import math

from model import Model
from independent_variable import IndependentVariable as IV
10  from witschey.base import memo_sqrt

class ZDT3(Model):

15      def __init__(self, ivs=30):

        def g(xs):
            return 1 + 9 * sum(xs[1:]) / (len(xs) - 1)

20        def f1(xs):
            return xs[0]

        def f2(xs):
            gxs = g(xs)
            a = 1 - memo_sqrt(xs[0] / gxs) - (xs[0] / gxs)
25            a *= math.sin(10 * math.pi * xs[0])
            return gxs * a

        ivs = tuple(IV(min=0, max=1) for _ in xrange(30))
30        super(ZDT3, self).__init__(independents=ivs, dependents=(f1, f2, g))

```

Sep 26, 14 21:59

"csc710sbse: hw4: Witschey"

Page 1/1

```

from simulated_annealer import SimulatedAnnealer ; del simulated_annealer
from maxwalksat import MaxWalkSat ; del maxwalksat

del searcher

```

Sep 30, 14 1:35

"csc710sbse: hw4: Witschey"

Page 1/1

```

from __future__ import division, unicode_literals

from witschey.base import memo, The

5 from datetime import datetime

class Searcher(object):

    def __new__(cls, *args, **kwargs):
10         # construct our object
        future_self = super(Searcher, cls).__new__(cls, *args, **kwargs)

        name = cls.__name__
        # initialize a dict with searcher's name
15         # and the initialization time
        d = dict(searcher=name, initialized=datetime.now())

        # if there are global options for this class or its bases in The
        for k in [name] + [k.__name__ for k in cls.__bases__]:
20             if hasattr(The, k):
                # add them to the dict
                d.update(getattr(The, k).__dict__)

        # then, add the kwargs to the constructor call to the dict.
25         # NB: this happens after adding options from The, so
        # call-specific options override the globals
        d.update(kwargs)

        # set our spec with the contents of the dict
30         future_self.spec = memo(**d)

        return future_self

    def __init__(self, model, *args, **kw):
35         self.model = model

    def run(*args, **kwargs):
        raise NotImplementedError()

```

Oct 06, 14 16:21

"csc710sbse: hw4: Witschey"

Page 1/2

```

from __future__ import division

import random
import numpy as np
5 from collections import defaultdict

from searcher import Searcher
from witschey.base import memo, tuple_replace
from witschey.log import NumberLog

10 class MaxWalkSat(Searcher):

    def __init__(self, model, *args, **kw):
        super(MaxWalkSat, self).__init__(model=model, *args, **kw)

15     def local_search_inputs(self, bottom, top, n=10):
        chunk_length = (top - bottom) / n

        for a in np.arange(bottom, top, chunk_length):
20             yield random.uniform(a, a + chunk_length)

    def run(self, text_report=True):
        rv = memo(report='')

25         log_eras = self.spec.log_eras or self.spec.terminate_early
        self.lives = 4

        if log_eras:
            rv.era_logs_by_objective = {f.__name__: defaultdict(NumberLog)
30                                     for f in self.model.ys}
            rv.era_logs_best_energy = defaultdict(NumberLog)

    def report(s):
        if text_report:
35             rv.report += s

        self.terminate = False
    def end_era(evals, era_length, log_value):
        report('\n{: .2}'.format(log_value) + ' ')
40
        self.lives -= 1
        eras = evals // era_length

45
        for logs in rv.era_logs_by_objective.values():
            if eras not in logs: break
            if len(logs.keys()) < 2: break

            prev_log = logs[logs.keys().index(eras) - 1]
50             if logs[eras].better(prev_log): self.lives += 1

            if self.lives <= 0: self.terminate = True

55
    def log_era(evals, era_length, dependents_outputs):
        era = evals // era_length
        for f, v in dependents_outputs:
            rv.era_logs_by_objective[f.__name__][era] += v
60             rv.era_logs_best_energy[era] += rv.best

        init = self.model.random_input_vector()
        solution = init
65         state = solution
        current_energy = self.model.energy(self.model(state))
        rv.best = current_energy
        evals = 0

70         report('{: .2}'.format(rv.best) + ' ')

        while evals < self.spec.iterations:

```

Oct 06, 14 16:21

"csc710sbse: hw4: Witschey"

Page 2/2

```

75         if self.terminate: break

        for j in range(20):
            if evals > self.spec.iterations or self.terminate:
                break

80         dimension = random.randint(0, len(state) - 1)
        if self.spec.p_mutation > random.random():
            state = tuple_replace(state,
                                dimension, self.model.xs[dimension]())

85         current_energy = self.model.energy(self.model(state))

        if current_energy < rv.best:
            solution = state
            rv.best = current_energy
            report('+')
90         else:
            report('.')

        evals += 1

95         if evals % self.spec.era_length == 0:
            end_era(evals, self.spec.era_length, rv.best)

100        else:
            for j in self.local_search_inputs(
                self.model.xs[dimension].min,
                self.model.xs[dimension].max
            ):
105                if self.terminate: break

            state = tuple_replace(state,
                                dimension, self.model.xs[dimension]())

110            current_energy = self.model(state)

            if current_energy < rv.best:
                solution = state
                rv.best = current_energy
                report('|')
115            else:
                report('.')

            evals += 1
            if evals % self.spec.era_length == 0:
                end_era(evals, self.spec.era_length, rv.best)

            if log_eras:
                log_era(evals, self.spec.era_length,
                        zip(self.model.ys, self.model(solution)))

125        rv.evaluations = evals
        return rv

```

Oct 06, 14 16:27

"csc710sbse: hw4: Witschey"

Page 1/2

```

from __future__ import division

import random
import math
5 from collections import defaultdict

from searcher import Searcher
from witschey.base import memo
from witschey.log import NumberLog
10

class SimulatedAnnealer(Searcher):
    def __init__(self, model, *args, **kw):
        super(SimulatedAnnealer, self).__init__(model=model, *args, **kw)

15    def run(self, text_report=True):
        rv = memo(report='')
        if self.spec.log_eras:
            rv.era_logs_by_objective = {
                f.__name__: defaultdict(NumberLog)
                for f in self.model.ys
            }
            rv.era_logs_best_energy = defaultdict(NumberLog)
        def report_append(s):
20            if text_report:
                rv.report += s

        init = self.model.random_input_vector()
        solution = init
        state = solution
        rv.best = self.model.energy(self.model(solution))

        def p(old, new, temp):
30            """
            sets the threshold we compare to to decide whether to jump

            returns e^(-(new-old)/temp)
            """
            numerator = new - old

            if not 0 <= numerator <= 1:
                numerator = old - new
            try:
                exponent = numerator / temp
            except ZeroDivisionError:
                return 0
            rv = math.exp(-exponent)
            if rv > 1:
                raise ValueError('p returning greater than one',
50                 rv, old, new, temp)
            return rv

        report_append('{: .2}'.format(rv.best) + ' ')
        self.lives = 4

55        for k in range(self.spec.iterations):
            if self.lives <= 0: break
            neighbor_candidate = self.model.random_input_vector()
            neighbor = tuple(neighbor_candidate[i]
                            if random.random() < self.spec.p_mutation else v
                            for i, v in enumerate(state))

            rv.best = self.model.energy(self.model(solution))
            neighbor_energy = self.model.energy(self.model(neighbor))
65            current_energy = self.model.energy(self.model(state))

            if neighbor_energy < rv.best:
                solution = neighbor
                rv.best = neighbor_energy
                report_append('!!')

70            if neighbor_energy < current_energy:

```

Oct 06, 14 16:27

"csc710sbse: hw4: Witschey"

Page 2/2

```

75         state = neighbor
           report_append('+')
       else:
           good_idea = p(
               self.model.normalize(current_energy),
               self.model.normalize(neighbor_energy),
80               k / self.spec.iterations)
           if good_idea < random.random():
               state = neighbor
               report_append('?')

85         report_append('.')

       if self.spec.log_eras or self.spec.terminate_early:
           era = k // self.spec.era_length
           for f, v in zip(self.model.ys, self.model(solution)):
90               rv.era_logs_best_energy[era] += rv.best
               rv.era_logs_by_objective[f.__name__][era] += v

       if k % self.spec.era_length == 0 and k != 0:
           report_append('\n' + '{: .2}'.format(rv.best) + ' ')

95       self.lives -= 1
           eras = k // self.spec.era_length

       for logs in rv.era_logs_by_objective.values():
100           if eras not in logs: break
               if len(logs.keys()) < 2: break

               prev_log = logs[logs.keys().index(eras) - 1]
               if logs[eras].better(prev_log): self.lives += 1

105       return rv

```