

```

1 local help = [[
2 SAW2: best or rest multi-objective optimization.
3 (c) 2023 Tim Menzies, tim@leee.org
4 *I think the highest and lowest points are the important ones.
5 Anything else is just...in between.* ~ Jim Morrison
6
7 USAGE: lua saw2.lua [OPTIONS]
8
9 OPTIONS:
10 -b --bins max bins = 16
11 -s --seed random number seed = 10019
12 -S --somed number of nums to keep = 256
13
14 OPTIONS (other):
15 -f --file where to find data = ../etc/data/autos3.csv
16 -d --dukexit on error = false
17 -h --help show help = false
18 -g --go start up action = nothing
19
20 Usage of the works is permitted provided that this instrument is
21 retained with the works, so that any entity that uses the works is
22 notified of this instrument. DISCLAIMER:THE WORKS ARE WITHOUT WARRANTY. ]]
23
24 local function string2thing(x)
25 x = xmatch("%s?(-)%s?")
26 if x=="true" then return true elseif x=="false" then return false end
27 return math.tointeger(x) or tonumber(x) or x end
28
29 local the={}
30 help:gsub("%n ([-|]|%s+)%s+([-|]|%s+)%n)%s([%s+])",function(f1,f2,k,x)
31 for n,flag in ipairs(arg) do if flag==f1 or flag==f2 then
32 x = x=="false" and"true" or x=="true" and"false" or arg[n+1] end end
33 the[k] = string2thing(x) end)
34
35 local any,atom,csv,has,many,map,merge,o,oo,obj,ok
36 local part,patch,per,push,rows,same,slice,sort
37 local GO,RANGE,SOME,NUM,SYM,COLS,ROW,EGS
38 local R,big,fmt
39
40 big = math.huge
41 R = math.random
42 fmt = string.format
43
44 function same(x) return x end
45 function push(t,x) t[1+#t]=x; return x end
46 function sort(t,f,u) table.sort(#t>0 and t or map(t,same), f); return t end
47 function map(t,f,u) u={}; for k,v in pairs(t) do u[1+#u]=f(v) end;return u end
48 function slice(t,i,j,k,u)
49 u={}; for n=(i or 1), (j or #t),(k or 1) do u[1+#u] = t[n] end return u end
50
51 function has(i, defaults, also)
52 for k,v in pairs(defaults) do i[k] = v end
53 for k,v in pairs(also or {}) do assert(i[k]==nil,"unknown:".k);i[k]=v end end
54
55 function csv(src)
56 src = io.input(src)
57 return function(line, row)
58 line=io.read()
59 if not line then io.close(src) else
60 row={}; for x in line:gmatch("(^[^,]+)") do row[1+#row]=string2thing(x) end
61 return row end end end
62
63 function oo(t) print(o(t)) end
64 function o(t, u)
65 if #t>0 then return "%.table.concat(map(t,tostiring),"").." else
66 u={}; for k,v in pairs(t) do u[1+#u] = fmt("%s%s",k,v) end
67 return (t.is or "")..("%.table.concat(sort(u),"").." end
68
69 function obj(name, t,new)
70 function new(kl,...)
71 local x=metatable({},{kl}); kl:new(x,...); return x end
72 t = {__tostiring=o, is=name or ""; t.__index=t
73 = t
74 return setmetatable(t, (__call=new)) end
75
76 RANGE=obj"RANGE"
77 function _new(i,t) has(i,(at=0, txt="", lo=big, hi= -big, ys=SYM()),t) end
78 function _of(i,x) return i.ys.all[x] or 0 end
79 function _lt(i,j) return i.lo < j.lo end
80 function _add(i,x,y)
81 if x=="?" then return x end
82 if x>i.hi then i.hi=x end
83 if x<i.lo then i.lo=x end
84 i.ys:add(y) end
85
86 function _select(i,t, x)
87 t = t.cells and t.cells or t
88 x = t[i.pos]
89 return x=="?" or i.lo == i.hi and i.lo == x or i.lo <= x and x < i.hi end
90
91 function _._tostring(i)
92 local x,o,hi = i.txt, i.lo, i.hi
93 if lo == hi then return fmt("%s==%s",x, lo)
94 elseif hi == big then return fmt("%s>=%s",x, lo)
95 elseif lo == -big then return fmt("%s<=%s", x, hi)
96 else return fmt("%s<=%s<%s",lo,x,hi) end end
97
98 function _merged(i,j,n0, k)
99 if i.at == j.at then
100 k = i.ys:merged(j.ys,n0)
101 if k then
102 return RANGE(at=i.at, txt=i.txt, lo=i.lo, hi=j.hi, ys=k) end end end
103
104 SOME=obj"SOME"
105 function _new(i) i.all, i.ok, i.n = {}, false,0 end
106 function _nums(i) i.all=i.ok and i.all or sort(i.all);i.ok=true;return i.all end
107
108 function _add(i,x)
109 if x=="?" then return x end
110 i.n = 1 + i.n
111 if #i.all < the.some then i.ok=false; push(i.all,x)
112 elseif R() < the.some/i.n then i.ok=false; i.all[R(#i.all)]=x end end
113
114 function _per(i,p, a)
115 a = i:nums()
116 return a[math.max(1, math.min(#a, (p or .5)*#a//1))] end

```

```

117
118 SYM=obj"SYM"
119 function _new(i,t) has(i,(at=0, txt="", n=0, all={}),t) end
120 function _add(i,x,n)
121 if x=="?" then n=n or 1; i.n=i.n+n; i.all[x]=n+(i.all[x] or 0) end end
122
123 function _mid(i, m,x)
124 m=0; for y,n in pairs(i.all) do if n>m then m,x=n,y end end; return x end
125
126 function _div(i, n,e)
127 e=0; for k,n in pairs(i.all) do e=e+n/i.n*math.log(n/i.n,2) end ;return e end
128
129 function _merge(i,j,n0, k)
130 k = SYM[at=i.at, txt=i.txt]
131 for x,n in pairs(i.all) do k:add(x,n) end
132 for x,n in pairs(j.all) do k:add(x,n) end
133 return f end
134
135 function _merged(i,j,n0, k)
136 n0, k = (n0 or 0), i:merge(j)
137 if i.n<n0 or j.n<n0 or
138 (i:div() * i.n+j:div() * j.n)/k.n > k:div() then return k end end
139
140 function _discretize(i,x,bins) return x end
141
142 NUM=obj"NUM"
143 function _new(i,t)
144 has(i,(at=0,txt="",lo= big,hi= -big, all=SOME()),t)
145 i.w = i.txt:find"5" and -1 or 1 end
146
147 function _mid(i) return i.all:per(.5) end
148 function _div(i) return (i.all:per(.9) - i.all:per(.1)) / 2.56 end
149 function _norm(i,x) return x=="?" and x or (x-i.lo)/(i.hi - i.lo) end
150
151 function _add(i,x)
152 if x=="?" then return x end
153 if x>i.hi then i.hi=x end
154 if x<i.lo then i.lo=x end
155 i.all:add(x) end
156
157 function _discretize(i,x,bins, base)
158 base = (i.hi - i.lo)/bins; return math.floor(x/base + 0.5)*base end
159
160 ROW=obj"ROW"
161 function _new(i,t) has(i,(cells={},backdrop={}),t) end
162
163 function _._lt(i,j, s1,s2,e,y,a,b)
164 y = i.backdrop.cols.y
165 s1, s2, e = 0, 0, math.exp(1)
166 for _col in pairs(y) do
167 a = col:norm(i.cells[col.at])
168 b = col:norm(j.cells[col.at])
169 s1= s1 - e^(col.w * (a - b) / #y)
170 s2= s2 - e^(col.w * (b - a) / #y) end
171 return s1/#y < s2/#y end
172
173 COLS=obj"COLS"
174 function _new(i,t, col)
175 has(i,{all={}, x={}, y={}, names={}),t)
176 for at,txt in pairs(i.names) do
177 col = push(i.all, (txt:find"^[A-Z]" and NUM or SYM)(at=at, txt=txt))
178 if not txt:find"5" then
179 push(txt:find"^[0-9]" and i.y or i.x, col) end end end
180
181 EGS=obj"EGS"
182 function _new(i) i.rows, i.cols= {},nil end
183 function _file(i,file) for row in csv(file) do i:add(row) end; return i end
184 function _add(i,row)
185 if i.cols
186 then row = push(i.rows, row.cells and row or ROW(backdrop=i, cells=row)).cells
187 for k,col in pairs(i.cols.all) do col:add(row[col.at]) end
188 else i.cols = COLS(names=row) end
189 return i end
190
191 function _mid(i,cs) return map(cs or i.cols.y,function(c)return c:mid() end)end
192 function _div(i,cs) return map(cs or i.cols.y,function(c)return c:div() end)end
193
194 function _copy(i,rows, out)
195 out=EGS(); add(i.cols.names)
196 for _row in pairs(rows or {}) do out:add(row) end
197 return out end
198
199 local _merge, _xpad, _ranges
200 function _ranges(i,one,two, t)
201 t={}; for _c in pairs(i.cols.x) do t[c.at]=_ranges(c,one,two) end;return t end
202
203 function _ranges(col,yes,no, out,x,d)
204 out = {}
205 for _what in pairs((rows=yes, klass=true), {rows=no, klass=false}) do
206 for _row in pairs(what.rows) do x = row.cells[col.at]; if x=="?" then
207 d = col:discretize(x,the.bins)
208 out[d] = out[d] or RANGE(at=col.at,txt=col.txt,lo=x,hi=x)
209 out[d]:add(x, what.klass) end end end
210 return _xpad(_merge(sort(out))) end
211
212 function _merge(b4, a,b,c,j,n,tmp)
213 j,n,tmp = 1,#b4,{ }
214 while j<=n do a, b = b4[j], b4[j+1]
215 if b then c = a:merged(b); if c then a,j = c,j+1 end end
216 tmp[#tmp+1] = a
217 j = j+1 end
218 return #tmp==#b4 and tmp or _merge(tmp) end
219
220 function _xpad(t)
221 for j=2,#t do t[j].lo=t[j-1].hi end; t[1].lo, t[#t].hi= -big,big; return t end

```

```

222
223 GO=obj"GO"
224 function ok(test,msg)
225 print("", test and "PASS" or "FAIL", msg or "")
226 if not test then
227 GO.fail= GO.fail+1
228 if the.dump then assert(test,msg) end end end
229
230 function _new(i,todo, defaults,go)
231 defaults={}; for k,v in pairs(the) do defaults[k]=v end
232 go={}; for k,_ in pairs(GO) do
233 if k=="new" and type(GO[k])=="function" then go[1+#go]=k end end
234 GO.fail= 0
235 for _x in pairs(todo=="all" and sort(go) or (todo)) do
236 for k,v in pairs(defaults) do the[k]=v end
237 math.randomseed(the.seed)
238 if GO[x] then print(x); GO[x]() end end
239 GO.rogue()
240 os.exit(GO.fail) end
241
242 function GO.rogue( t)
243 t={}; for _k in pairs( _G*, _VERSION*, "arg", "assert", "collectgarbage",
244 "coroutine", "debug", "dofile", "error", "getmetatable", "io", "ipairs",
245 "load", "loadfile", "math", "next", "os", "package", "pairs", "pcall",
246 "print", "rawequal", "rawget", "rawlen", "rawset", "require", "select",
247 "setmetatable", "string", "table", "tonumber", "tostiring", "type", "utf8",
248 "warn", "xpcall" do t[k]=k end
249 for k,v in pairs(_ENV) do if not t[k] then print("?",k, type(v)) end end end
250
251 function GO.cols()
252 oo(COLS(names={"Cylids", "Acc*"})) end
253
254 function GO.egs( egs,n)
255 egs = EGS():file(the.file)
256 sort(egs.rows)
257 print("all", o(egs:mid()))
258 n = (#egs.rows)^.5 // 1
259 print("best",o(egs:copy(slice(egs.rows,1,n):mid())))
260 print("rest",o(egs:copy(slice(egs.rows,n+1):mid()))) end
261
262 function GO.egs1( egs,best,rest,n)
263 egs = EGS():file(the.file)
264 sort(egs.rows)
265 n = (#egs.rows)^.5 // 1
266 best = slice(egs.rows, 1, n)
267 rest = slice(egs.rows, n+1, #egs.rows, (#egs.rows - n)//(3*n))
268 print("all", o(egs:mid()))
269 print("best",o(egs:copy(best):mid()))
270 print("rest",o(egs:copy(rest):mid()))
271 egs:ranges(best,rest)
272 end
273
274 --xxx replace part with a slice wit one extra art
275 function part(t,n,lo,hi, u)
276 lo, hi = 1, hi or #t
277 u={};for j = lo, hi, (hi-lo)//n do push(u,t[j]) end; return u end
278
279 if the.help
280 then print(help:gsub("%u%u+", "%U7[3lm%]\27[0m]")
281 help:gsub("(%s)([-|]|%s+)%s+", "%\27[33m%227[0m%3*),"")
282 else GO(the.go) end
283
284
285
286
287
288
289
290
291
292
293

```

"This ain't chemistry.  
This is art."