In [ ]:
```
using Pkg
Pkg.add https://github.com/baggepinnen/Hyperopt.jl.git
Pkg.add https://github.com/mohamed82008/Nonconvex.jl.git
Pkg.add https://github.com/SciML/GalacticOptim.jl.git
```

In [33]:
```
using Hyperopt,GalacticOptim
```

The function optimized should be calling another optimizer from GalacticOptim.

In the introduction of GalacticOptim, I find many "Local Gradient-Based Optimization","Local Derivative-Free Optimization", "Local

Hessian-Based Second Order Optimization" "Local Hessian-Free Second Order Optimization", "Global Unconstrained Optimizers"methods.

And I also find a function in Hyperopt named "Hyperband" method can be used to call the local or global optimizer from GalacticOptim.

For example, we want to optimize the function f, and we can use SimulatedAnnealing() as our local optimizing function.

We adopt one local optimizer ParticleSwarm() as an example:

In [51]:
```
using Optim
f(a;c=10) = sum(@. 100 + (a-3)^2 + (c-100)^2)
hohb = @hyperopt for i=100, sampler=Hyperband(R=50, η=3, inner=RandomSampler()), a = LinRan
    if !(state === nothing)
        a,c = state
    end
    res = Optim.optimize(x->f(x[1],c=x[2]), [a,c], ParticleSwarm(), Optim.Options(f_calls_lim
    @show Optim.minimum(res), Optim.minimizer(res)
end
```

```
(Optim.minimum(res), Optim.minimizer(res)) = (3302.942420444705, [6.828991009511888, 43.
535132619532504])
(Optim.minimum(res), Optim.minimizer(res)) = (5584.67295627623, [8.957450564377947, 26.1
81426869322231])
(Optim.minimum(res), Optim.minimizer(res)) = (6414.154374818459, [9.789082960369441, 20.
828902196725352])
(Optim.minimum(res), Optim.minimizer(res)) = (2891.8375164503564, [9.549280406424725, 4
7.5696229022871])
(Optim.minimum(res), Optim.minimizer(res)) = (7446.940991777658, [6.640193600833616, 14.
36303378606813])
(Optim.minimum(res), Optim.minimizer(res)) = (8000.911258703172, [3.908191906298828, 11.
117569530505607])
(Optim.minimum(res), Optim.minimizer(res)) = (101.39011073103032, [3.881815320112066, 9
9.21736823968997])
(Optim.minimum(res), Optim.minimizer(res)) = (150.25254230465737, [3.999731122526228, 9
2.9819461396118])
(Optim.minimum(res), Optim.minimizer(res)) = (150.03263088697582, [2.6813397378587753, 9
2.9338067869391])
(Optim.minimum(res), Optim.minimizer(res)) = (100.61643567052181, [3.062635439455999, 9
9.21736823968997])
```

We can also call and optimize multiple optimizer to optimize one function and we use Hyperopt to optimize multiple local optimizer.

For example, in the code:

algorithm = [LBFGS(),SimulatedAnnealing(),ParticleSwarm(), NelderMead(), BFGS(), NewtonTrustRegion()], we use three of them.

In [53]:
```
hohb = @hyperopt for i=10, sampler=Hyperband(R=50, η=3, inner=RandomSampler()),
    algorithm = [LBFGS(),SimulatedAnnealing(),ParticleSwarm()],
    a = LinRange(1,5,1800),
    c = exp10.(LinRange(-1,3,1800))
    if !(state === nothing)
        x0,algorithm = state
    else
        x0 = [a,c]
    end
    println(i, " algorithm: ", typeof(algorithm).name.name)
    res = Optim.optimize(x->f(x[1],c=x[2]), x0, algorithm, Optim.Options(time_limit=i+1, show
    Optim.minimum(res), (Optim.minimizer(res), algorithm)
end
```

```
5353314332  …  954.9681399564245, 959.8698272584649, 964.7966740788197, 969.74880955697
17, 974.7263634952538, 979.7294663622529, 984.7582492962279, 989.8128441085481, 994.8933
83287148, 1000.0]), Any[([3.0000000520148795, 99.99999997527875], ParticleSwarm{Any}(Any
[], Any[], 0)), ([3.000000000010182, 100.00000000000081], LBFGS{Nothing, LineSearches.In
itialStatic{Float64}, LineSearches.HagerZhang{Float64, Base.RefValue{Bool}}, Optim.var"#
17#19"}(10, LineSearches.InitialStatic{Float64}
  alpha: Float64 1.0
  scaled: Bool false
, LineSearches.HagerZhang{Float64, Base.RefValue{Bool}}
  delta: Float64 0.1
  sigma: Float64 0.9
  alphamax: Float64 Inf
  rho: Float64 5.0
  epsilon: Float64 1.0e-6
  gamma: Float64 0.66
  linesearchmax: Int64 50
  psi3: Float64 0.1
  display: Int64 0
  mayterminate: Base.RefValue{Bool}
  nothing  Optim.var"#17#19"()  Flat()  true))  ([3.0066203040385835  100.0268464060837
```