Expensive Multiobjective Optimization via MOEA/D

Qingfu Zhang
Department of Computer Science,
City University of Hong Kong,
Hong Kong

Outline

GP Modelling for Single Obj Expensive Opt and Fuzzy Clustering

Problem

Consider

$$\min g(x)$$

where $x = (x_1, \ldots, x_n) \in D \subset R^n$ and g is continuous of x.

- g-function evaluation=computer/physical experiments.
- g is black-box and its evaluation is extremely expensive.
- The computational budget could be very limited, say, at most 200 g-function evaluations.

Task

To find a reasonably good solution to

$$\min g(x)$$

with a small number of function evaluations?

- Black-box $\Rightarrow$ no math formulation: Impossible to use a simple math function to approximate it. Traditional math programming methods do not work.
- Expensive function evaluation $\Rightarrow$ Traditional heuristics (evolutionary methods) do not work, either.

Surrogate Model (Response Surface) Method

- Step 0 **Initialization**: Carefully select a small number of points from the x-space and evaluate their g-function values.
- Step 1 **Modeling:** Based on **All** the evaluated g-function values, build a surrogate model of g.
- Step 2 **Locating new test points:** Based on the surrogate model, predict the most promising new points.
- Step 3 **Function evaluation:** Evaluate these new test points. If the stopping condition is not met, go to Step 1. Otherwise, output the best point found so far.

Surrogate Models: Neural Networks, Radial basis function, **Gaussian Process (GP) Model**....

Gaussian Process (GP) Modeling

Assumption:

To build a cheap surrogate model for $y = g(x)$, $x \in R^n$, assumes

- For any x, $g(x)$ is a sample of

$$\mu + \epsilon(x) \tag{1}$$

where $\epsilon(x) \sim N(0, \sigma^2)$, μ and σ are independent of x.

- For $x, x' \in R^n$, $c(x, x')$, the correlation btw $\epsilon(x)$ and $\epsilon(x')$:

$$c(x, x') = \exp[-d(x, x')], \tag{2}$$

where $d(x, x') = \sum_{i=1}^{n} \theta_i |x_i - x_i'|^{p_i}$. $\theta_i > 0$ and $1 \leq p_i \leq 2$.

$\theta_i > 0$ and $1 \leq p_i \leq 2$.

$c(x, x') \nearrow 1$ as $d(x, x') \searrow 0$.

implies $g(x') \to g(x)$ as $d(x, x') \searrow 0$. It means that $g(x)$ is continuous.

Two things to do:

Parameter estimation of $\theta_1, \ldots, \theta_n$, $p_1, \ldots, p_n$, μ and σ. ($2n + 2$ parameters).

Building posterior predictive models for predicting $g(x)$ at untested x.

Parameter Estimation

- Given K points $x^1, \ldots, x^K \in R^n$ and their g-function values $y^1, \ldots, y^K$,
- the parameters μ, σ, $\theta_1, \ldots, \theta_n$, and $p_1, \ldots, p_n$ can be estimated by maximizing the likelihood that $g(x) = y^i$ at $x = x^i$ $(i = 1, \ldots, K)$:

$$\frac{1}{(2\pi\sigma^2)^{K/2}\sqrt{det(C)}} \exp\left[-\frac{(y - \mu\mathbf{1})^T C^{-1}(y - \mu\mathbf{1})}{2\sigma^2}\right] \qquad (3)$$

where C is a $K \times K$ matrix whose (i,j)-element is $c(x^i, x^j)$, $y = (y^1, \ldots, y^K)^T$ and $\mathbf{1}$ is a K-D column vector of ones.

Remarks:

- $\theta_1, \ldots, \theta_n$, and $p_1, \ldots, p_n$ are in $c(x^i, x^j)$,
- maximization of (3) is not costly, it involves determinants and inverse.

GP Predictive Distribution

Given hyper parameters θ_i, p_i, μ and σ^2. Under the condition $g(x^i) = y^i$ for $i = 1, \ldots, K$, for any $x \in R^n$, the conditional probability of $g(x)$ is:

$$N(\hat{y}(x), s^2(x))$$

where

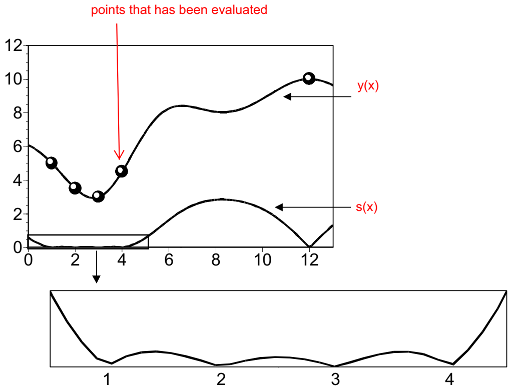
$$\hat{y}(x) = \mu + r^T C^{-1}(y - \mathbf{1}\mu) \tag{4}$$

$$s^2(x) = \sigma^2[1 - r^T C^{-1} r + \frac{(1 - \mathbf{1}^T C^{-1} r)^2}{\mathbf{1}^T C^{-1} r}] \tag{5}$$

where $r = (c(x, x^1), \ldots, c(x, x^K))^T$.

- different untested points have different predict models.
- $s^2(x)$ measures the uncertainty.

Example: (D. Jones et al 1998)



(D. Jones 2001)

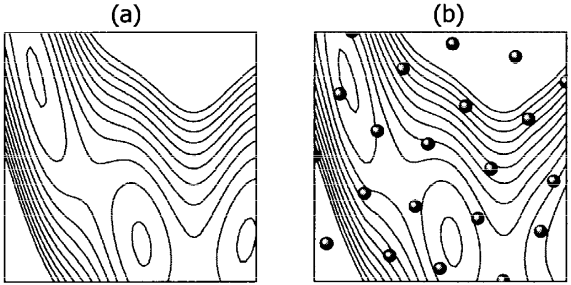


Figure 1. (a) Contours of the Branin test function. (b) Contours of a kriging surface fit to 21 points (shown as spheres).

How to build a GP predictive model

- Assume that $y = g(x)$ is a sample of a GP model
- Suppose K points $x^1, \dots, x^K \in R^n$ and their g-function values $y^1, \dots, y^K$ are given.
- Using the maximum likelihood estimation, estimate hyper parameters.
- Compute the conditional probability $N(\hat{y}(x), s^2(x))$ for $g(x)$ at untested point x.

Q: Suppose we have evaluated g-function value at $x^1, \ldots, x^K$ and the smallest function value among these K points is g_{min}. which point(s) should be evaluated next?

A: (D. Jones et al 1998) the point maximizing a utility function such as

$$E[I(x)] = E[\max\{g_{min} - g(x), 0\}] \quad \text{expected improvement,}$$

and

$$P(g(x) < g_{min}) = \Phi(\frac{g_{min} - \hat{y}(x)}{\hat{s}(x)}) \quad \text{prob. of improvement.}$$

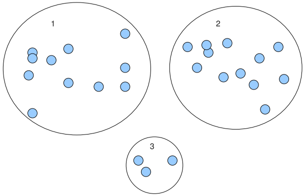
Methods using these utility functions are called **Efficient Global Optimization (EGO)**.

The major computational cost:
function evaluation + modeling (i.e., estimation of parameters) + maximization of EI (or PI).

When $K > 300$, the overhead of model building is too high. This method is impractical.

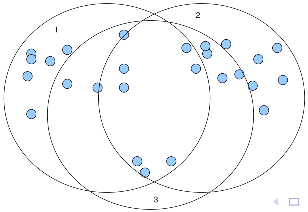
Clustering for large data set

- Commonly-used strategies for dealing with large K:
  - select a limited number of evaluated points.
    -: doesn't make the full use of all the evaluated points
  - do crisp clustering
    -: The prediction quality is poor in boundary areas.



Fuzzy-Clustering for a large data set

- Two parameters:
  - L: the number of points for building a local model.
  - c: the number of clusters.
- K evaluated points are clustered by the Fuzzy C-Means Clustering into c clusters with cluster centers $v^1, \ldots, v^c$.
- P_i is set to the set containing the L evaluated points with the highest membership degrees to cluster i.
- Using the data points in P_i, build GP model i.



Modelling Using Fuzzy-Clustering

- **Closest Prediction**: If v^k is the closest cluster center to x, then use GP model k for modeling $g(x)$:

$$N(\hat{y}(x)_k, s^2(x)_k)$$

- **Combination of Different Models**: This approach combines the predictions from all the local models:

$$N(\hat{y}(x), s^2(x))$$

where

$$\hat{y}(x) = \sum_i p_i \hat{y}(x)_i \quad s^2(x) = \sum_i (p_i s(x)_i)^2$$

p_i is the membership degree of x to cluster i.

MOEA/D+EGO for Multiobj Expensive Optimization

Problem

$$\begin{array}{ll} \text{minimize} & F(x) = (f_1(x), \ldots, f_m(x))^T \\ \text{subject to} & x = (x_1, \ldots, x_n)^T \in \prod_{i=1}^{n} [a_i, b_i] \end{array} \tag{6}$$

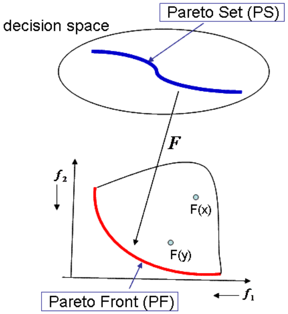
where

- all the f_i are continuous.
- $-\infty < a_i < b_i < +\infty$.
- m is small, $m = 2$ or 3.
- Function evaluation is **very expensive** .

There is no single optimal solution, one must **balance** different objs!

Pareto Optimal Solutions= Best Trade-off Candidates

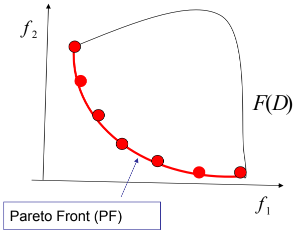
- y dominates x if y is no worse than x in any objs, and y is better than x in at least one obj.
- x is Pareto optimal iff no other solution dominates it.
- Pareto set (PS): all the Pareto optimal solution in the x-space.
- Pareto front (PF): the image of the PS in the f-space.



- no decision maker likes non-Pareto optimal solutions.
- a decision maker often want to have a number of well representative Pareto optimal solutions for making her final decision.

Task:

Find a small number (say, $10 - 20$) of well representative Pareto optimal solutions by using 100-300 function evaluations for problems with 2 or 3 objectives and less than 10 variables.



How to do EGO in multiobjective Opt

- consider a random aggregation function at each iteration (J. Knowles, 2006).
- generalize the expected improvement to MOPs (Keane, Emmerich et al, 2006).

All the above approaches can generate only one test point at each iteration.

Our Goal:

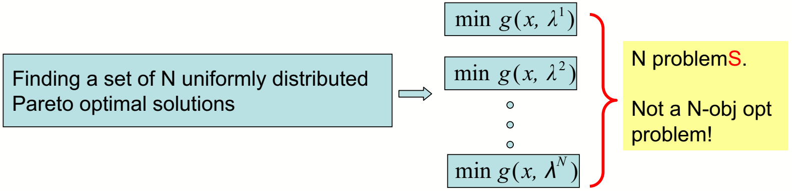
- to generate several new test points at each iteration for parallel computing.

Our Approach:

- MOEA/D+EGO

MOEA/D Principle

- Decomposition: Decompose the task of approximating the PF into N single objective subproblems. The optimal solutions of these subproblems form a good approximation to the PF.
- Collaboration: Optimize these subproblems in a collaborative manner.


Finding a set of N uniformly distributed Pareto optimal solutions
$\min\ g(x, \lambda^1)$
$\min\ g(x, \lambda^2)$
$\min\ g(x, \lambda^N)$
N problemS.
Not a N-obj opt problem!

Decomposition

Weighted Sum Approach

$$\text{minimize } g^{ws}(x,\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)$$

where $\lambda = (\lambda_1, \dots, \lambda_m)$ be a weight vector, i. e., $\sum_{i=1}^{m} \lambda_i = 1$ and all the $\lambda_i \geq 0$.

- If the PF is convex, then for any Pareto optimal solution x^*, there exists a weight vector such that x^* is the optimal solution to the above problem.
- This approach does not work for nonconvex PFs.

Techbycheff Approach

$$\text{minimize } g^{te}(x, \lambda) = \max_{1 \leq i \leq m} \{\lambda_i (f_i(x) - z_i^*)\}$$

where $z^* = (z_1^*, \ldots, z_m^*)$ is a reference point, i. e. $z_i^* < \min f_i$.

- for each Pareto optimal point x^* there exists a weight vector λ such that x^* is the optimal solution of the above problem.
- This approach can deal with nonconvex PFs

Ideas

- These subproblems are related with each other.
- subproblems with similar weight vectors have similar solutions.
- neighborhood relationships among all the subproblems can be defined.
- Explore these neighborhood relationships and solve these subproblems in a single run.

MOEA/D+EGO

Step 1 **Initialization**: Carefully generate a small number of points and evaluate them.

Step 2 **Models Building**: By using the evaluated function values, build a predictive model for each objective $g(x|\lambda^i)$ and then use it to define $\xi^i(x)$, a metric measuring the merit of evaluating point x for optimizing $g(x|\lambda^i)$.

Step 3 **Locating Candidate Points**: Using MOEA/D, obtain $\tilde{x}^1, \ldots, \tilde{x}^N$, where $\tilde{x}^i$ is an approximate solution for maximizing $\xi^i(x)$.

Step 4 **Selecting Points for Function Evaluation**: Select K_E points from $\tilde{x}^1, \ldots, \tilde{x}^N$ using a selection scheme.

Step 5 **Function Evaluations**: Evaluate the F-function values of all the K_E selected points in **Step 4**, then go to **Step 2**.

Model building in MOEA/D-EGO

Model building is very costly. How can one reduce the cost? Solution:

- build predictive model $N(\hat{y}_i(x), \hat{s}_i^2(x))$ for each individual obj function f_j by maximum likelihood estimation.
- assume that $f_1, \ldots, f_m$ are independent of each other.
- mathematically induce the predictive models for all the subproblems.

In the case of weighted sum approach

Since

$$g^{ws}(x,\lambda)=\sum_{i=1}^{m}\lambda_i f_i(x),$$

Its predictive model can be $N(\hat{y}^{ws},(\hat{s}^{ws})^2)$ where

$$\hat{y}^{ws}=\sum_{i=1}^{m}\lambda_i\hat{y}_i(x),\quad (\hat{s}^{ws})^2=\sum_{i=1}^{m}[\lambda_i\hat{s}_i^2(x)]^2.$$

In the case of Techbycheff approach
some math tricks are needed. We have discussed about the cases when $m = 2, 3$.

The details can be found in the paper.

Selecting Points for Evaluations in MOEA/D-EGO

We have the following the considerations:

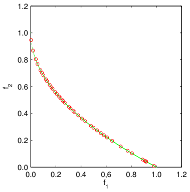
- The selected points should be as different as possible from those points already evaluated.
- The selected points should not be too close to each other.
- The selected points should have higher EI (PI)-values.

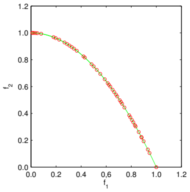
The details of the method can be found in the paper.

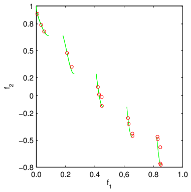
Experimental Settings

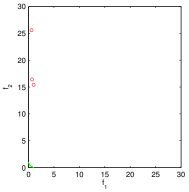
- Test problems, 2 objectives with 8 variables, 3 objectives with 6 variables.
- the number of function evaluations=200 for bi-obj problems, 300 for 3 obj problem.
- $(11n-1)$ initial test points are generated by Latin hypercube sampling method.
- at each generation, 5 points are selected for evaluation.
- Techebycheff approach are used.
- the number of subproblems=300 for two objs, 595 for three objs.

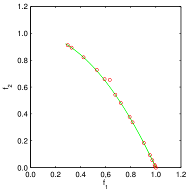
Results

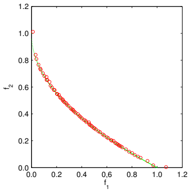

(a) ZDT1

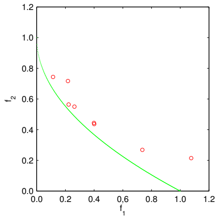

(b) ZDT2


(c) ZDT3

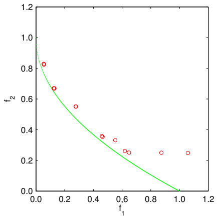

(d) ZDT4


(e) ZDT6

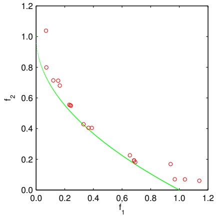

(f) LZ08-F1



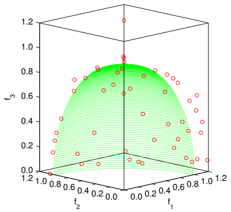
(g) LZ08-F2



(h) LZ08-F3



(i) LZ08-F4



(j) DTLZ2

Why some are poor?

One reason might be that these functions cannot be modeled by Gaussian process model, i.e. don't meet the assumption in modeling.

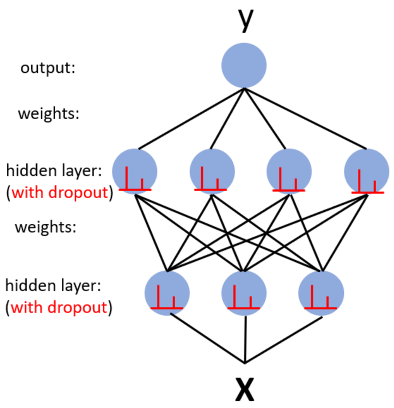
Use of Gradients and Dropout NN in MOEA/D for Expensive Multiobjective Optimization

Motivation

- GP modelling does not scale well. It is difficult to handle more than 20 variables, and a large data set with more than 1,000 points.
- In some real-life expensive optimization problems, gradients are available.

Goal: Design scalable MOEA/D which be able to use gradients.

Bayesian Neural Network with Monte Carlo Dropout

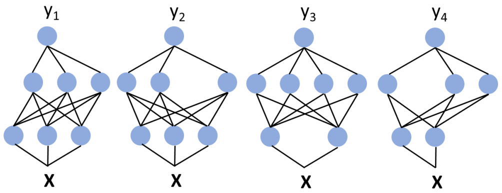


$$\hat{y}_{dropout} = h(xZ_1\bar{W}_1 + b_1)Z_1\bar{W}_2 + b_2$$
$$Z_1 = \text{diag}(z_1), Z_2 = \text{diag}(z_2)$$

$\bar{W}_i$ is the fixed weight matrix.
$z_{ik} \sim \text{Bernoulli}(p)$ is the dropout mask.

- $y = f(x, z)$
- y is a random variable as well.

Bayesian Neural Network with Monte Carlo Dropout



Randomly drop the nodes:

- fast on training/prediction.
- provide practical distribution estimation.

Bayesian Neural Network with Monte Carlo Dropout[1]

Prediction:

$$\mathbb{E}(\hat{y}) = \frac{1}{S}\sum_{s=1}^{S} \hat{y}_s(x), \tag{7}$$

$$\mathrm{Var}(\hat{y}) = \frac{1}{S}\sum_{s=1}^{S} [\hat{y}_s(x) - \mathbb{E}(\hat{y})]^2. \tag{8}$$

[1] Gal, Yarin, and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. ICML 2016.

Using Gradient Information: Sobolev Training[2]

$$
\begin{aligned}
L &= L_e + L_g \\
&= \sum_{i=1}^{N} l(\hat{f}(x_i|W), f(x_i)) + \sum_{i=1}^{N} l(\nabla \hat{f}(x_i|W), \nabla f(x_i)) \\
&= \sum_{i=1}^{N} [\hat{f}(x_i|W) - f(x_i)]^2 + \sum_{i=1}^{N} [(\nabla \hat{f}(x_i|W) - f(x_i))]^2 \qquad (9)
\end{aligned}
$$

- L_e: Error Loss
- L_g: Gradient Loss

[2] Czarnecki, Wojciech M., et al, Sobolev training for neural networks, NeurIPS 2017

Dropout NN with Gradient Information

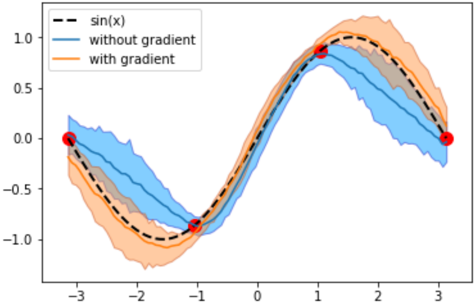


Figure: The predictive mean (solid line) ± two standard deviations (shade area) obtained by dropout NN with and without gradient information for sin(x).

Training Time

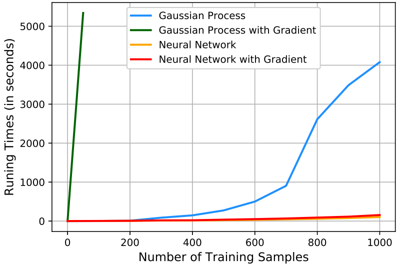
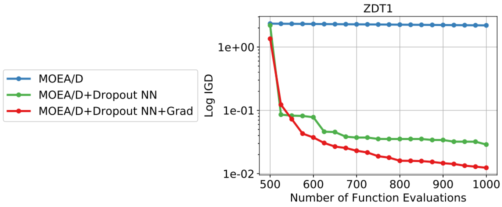
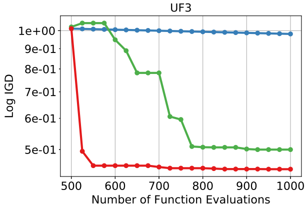


Figure: The training time of GP models and Dropout NN with and without gradients

Preliminary Experimental Results



(a) ZDT1



(b) UF3

Figure: Evolutions of the median IGD values obtained by different algorithms on problems with 50 decision variables and 1000 function evaluations.

Conclusion

Conclusion

- In expensive optimization. all the info obtained in the previous search should be used for determining the next test point.
- Fuzzy clustering can be used for improving scalability of GP modelling.
- MOEA/D+EGO works for small scale expensive optimization optimization.
- MOEA/D+Dropout NN can use gradient info and handle large scale problems. It is promising.
- A DM only needs one final solution, it is worthwhile studying how to do interaction with the DM to reduce the comoputational cost.

Ref

Q. Zhang, W. Liu, E. Tsang and B. Virginas, *Expensive Multiobjective Optimization by MOEA/D with Gaussian Process Model*, IEEE Trans on Evolutionary Computation, 2010.

X. Lin and H. Zhen and Z. Li and Q. Zhang and S. Kwong, *A Batched Scalable Multi-Objective Bayesian Optimization Algorithm*, https://arxiv.org/abs/1811.01323, 2018

Thank you!