Multi-objective Evolutionary Learning:
Advances in Theories and Algorithms

Chao Qian

School of Computer Science and Technology
University of Science and Technology of China

http://staff.ustc.edu.cn/~chaoqian/
Email: chaoqian@ustc.edu.cn

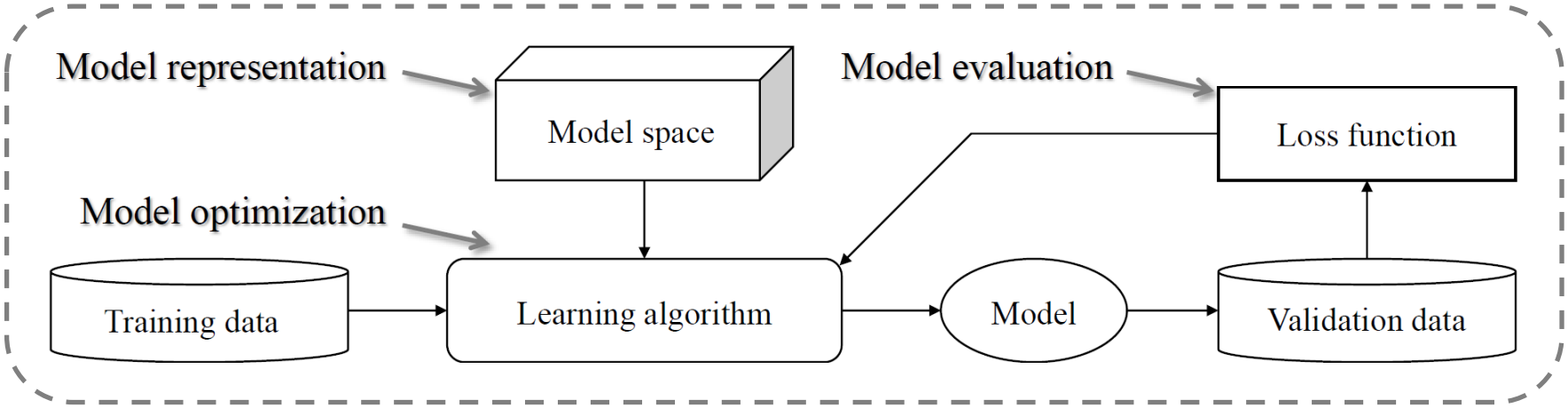
Outline

- **Introduction**
- Running time analysis approaches for MOEAs
- Theoretical properties of MOEAs
  - Constrained optimization, noisy optimization
- Multi-objective evolutionary learning algorithms
  - Selective ensemble, subset selection
- Conclusion

Machine learning

Machine learning aims at learning generalizable models from data

- Model representation, evaluation, optimization [Domingos, CACM'12]



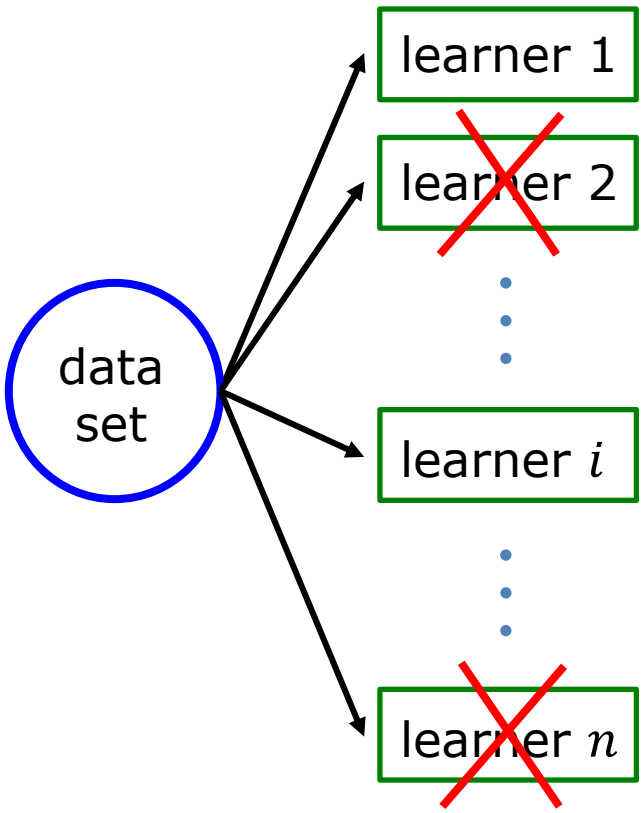
A machine learning problem ⇨ A complicated optimization problem

Non-unique objectives

Selective ensemble

Ensemble learning [Zhou, 2012]

- better performance than a single learner



Selective ensemble (ensemble pruning) [Zhou, 2012]

- better performance than the complete ensemble
- reduce storage and improve efficiency

Two goals

- maximize the generalization performance
- minimize the number of selected learners

Multi-objective machine learning

Machine learning tasks often involve multiple conflicting objectives

Selective ensemble [Zhou, 2012]

- maximize the generalization performance
- minimize the number of selected learners

Clustering [Jain & Dubes, 1988]

- maximize the intercluster similarity
- minimize the intracluster similarity

Active learning [Huang et al., TPAMI'14]

- informative
- representative
- diverse

How to solve multi-objective optimization problems efficiently?

Multi-objective optimization

The task: optimize multiple objectives simultaneously

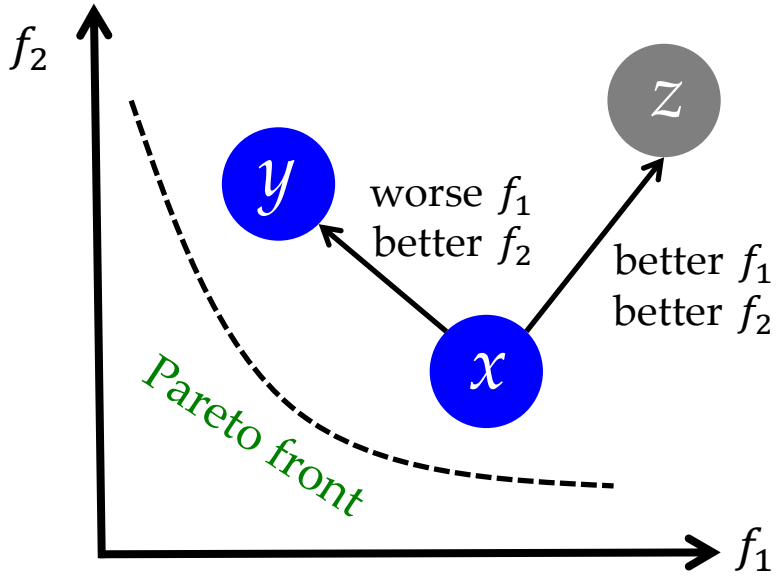
$$min_{x \in \mathcal{X}} \ (f_1(x), f_2(x), \ldots, f_m(x))$$

x dominates z :

$$f_1(x) < f_1(z) \wedge f_2(x) < f_2(z)$$

x is incomparable with y :

$$f_1(x) > f_1(y) \wedge f_2(x) < f_2(y)$$



Multi-objective optimization methods

- Convert into a single-objective optimization problem
  - linear scalarization:

$$min_{x \in \mathcal{X}} \quad w_1 f_1(x) + \cdots + w_m f_m(x)$$

  - ϵ-constraint method:

$$min_{x \in \mathcal{X}} \quad f_j(x) \quad s.t. \quad \forall i \neq j: \; f_i(x) \leq \epsilon_i$$

e.g., one optimization-based selective ensemble algorithm [Zhang et al., JMLR'06]:

$$min_{\boldsymbol{x}} \; \boldsymbol{x}^T \widetilde{\boldsymbol{G}} \boldsymbol{x} \quad s.t. \quad \sum_{i=1}^{N} x_i = T, x_i \in \{0,1\}$$

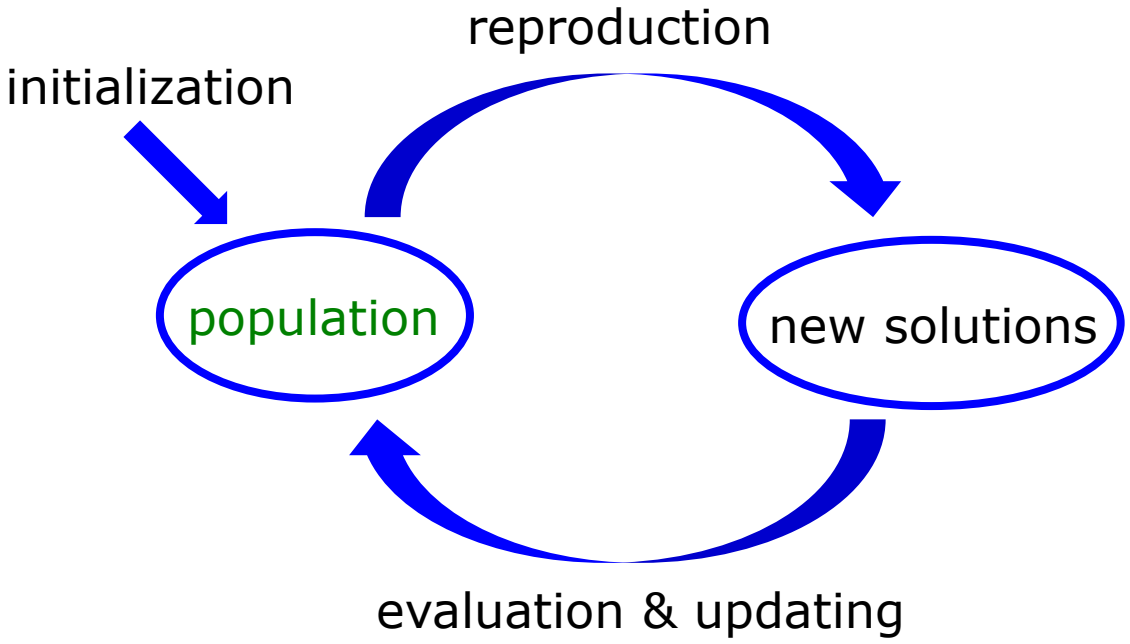
the error — the size

The coefficients w_i or ϵ_i is hard to determine, and only one solution is generated

Evolutionary algorithms

Evolutionary algorithms: a kind of nature-inspired randomized heuristic optimization algorithms

genetic algorithms, evolutionary strategies, evolutionary programming, particle swarm optimization, ……



generate multiple solutions in one run

⇩

multi-objective optimization

Multi-objective evolutionary algorithms

- Many successful multi-objective applications
  - engineering design [Coello Coello & Lamont, 2004]
  - medicine [Toro et al., TBME'06]
  - finance and economics [Ponsich et al., TEC'13]

- Multi-objective evolutionary algorithms
  - SPEA [Zitzler & Thiele, TEC'99]
  - NSGA-II [Deb et al., TEC'02]
  - MOEA/D [Zhang & Li, TEC'07]

- Advantages
  - generate multiple solutions in one run
  - not need to select proper coefficients before optimization

easier to select one solution after optimization

Multi-objective evolutionary learning

- MOEAs have been applied in machine learning
  - feature selection [Mukhopadhyay et al., TEC'14a]
  - clustering [Mukhopadhyay et al., TEC'14b]
  - multi-label learning [Shi et al., TIST'14]
  - active learning [Reyes & Ventura, TIST'18]
- MOEAs have yielded encouraging empirical outcomes, but lack theoretical support
- The theoretical understanding of MOEAs is underdeveloped

Outline

- Introduction
- **Running time analysis approaches for MOEAs**
- Theoretical properties of MOEAs
  - Constrained optimization, noisy optimization
- Multi-objective evolutionary learning algorithms
  - Selective ensemble, subset selection
- Conclusion

Running time analysis

Convergence analysis

$lim_{t\to+\infty} \mathrm{P}(\xi_t \in \mathcal{X}^*) = 1$?

Running time analysis

$\tau = \min\{t \geq 0 \mid \xi_t \in \mathcal{X}^*\}$

The leading theoretical aspect

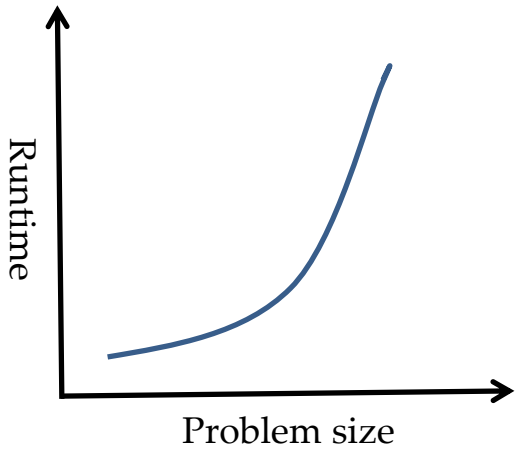
[Neumann & Witt, 2010; Auger & Doerr, 2011]

The number of iterations until finding an optimal or approximate solution for the first time

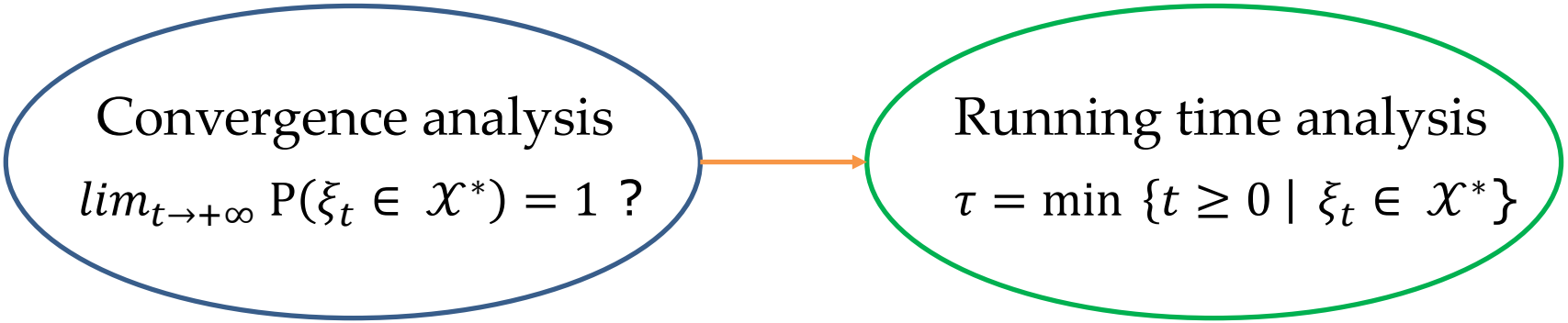
Running time complexity

- The number of iterations × the number of fitness evaluations in each iteration
- Usually grows with the problem size and expressed in asymptotic notations

  e.g., (1+1)-EA solving LeadingOnes: $O(n^2)$



Running time analysis



The leading theoretical aspect

[Neumann & Witt, 2010; Auger & Doerr, 2011]

The number of iterations until finding an optimal or approximate solution for the first time

A quick guide to asymptotic notations:

Let g and f be two functions defined on the real numbers.

- $g \in O(f)$: $\exists M > 0$ such that $g(x) \leq M \cdot f(x)$ for all sufficiently large x
- $g \in \Omega(f)$: $f \in O(g)$
- $g \in \Theta(f)$: $g \in O(f)$ and $g \in \Omega(f)$

$g \in O(f) \to g \leq f$

$g \in \Omega(f) \to g \geq f$

$g \in \Theta(f) \to g = f$

Running time analysis of MOEAs

- Running time analyses of MOEAs are rare and case-specific

	GSEMO		SEMO
synthetic problems	LOTZ: $O(n^3)$ [Giel, CEC'03]	COCZ: $O(n^2 \log n)$ [Qian et al., AIJ'13]	mLOTZ, mCOCZ: $O(n^{m+1})$ [Laumanns et al., TEC'04]

More results: [Friedrich et al., TCS'10; Giel & Lehre, ECJ'10; Friedrich et al., TCS'11; Neumann, GECCO'12; Doerr et al., CEC'13; GECCO'16; Qian et al., PPSN'16; Osuna et al., GECCO'17]

	GSEMO	a variant of GSEMO
combinatorial problems	bi-objective MST [Neumann, EJOR'07]	multi-objective shortest paths [Horoba, FOGA'09; Neumann & Theile, PPSN'10]

- Analyses starting from scratch are quite difficult
- Existing general approaches, e.g., fitness level [Sudholt, TEC'13] and drift analysis [He & Yao, AIJ'01], are hard to be applied directly

Switch analysis

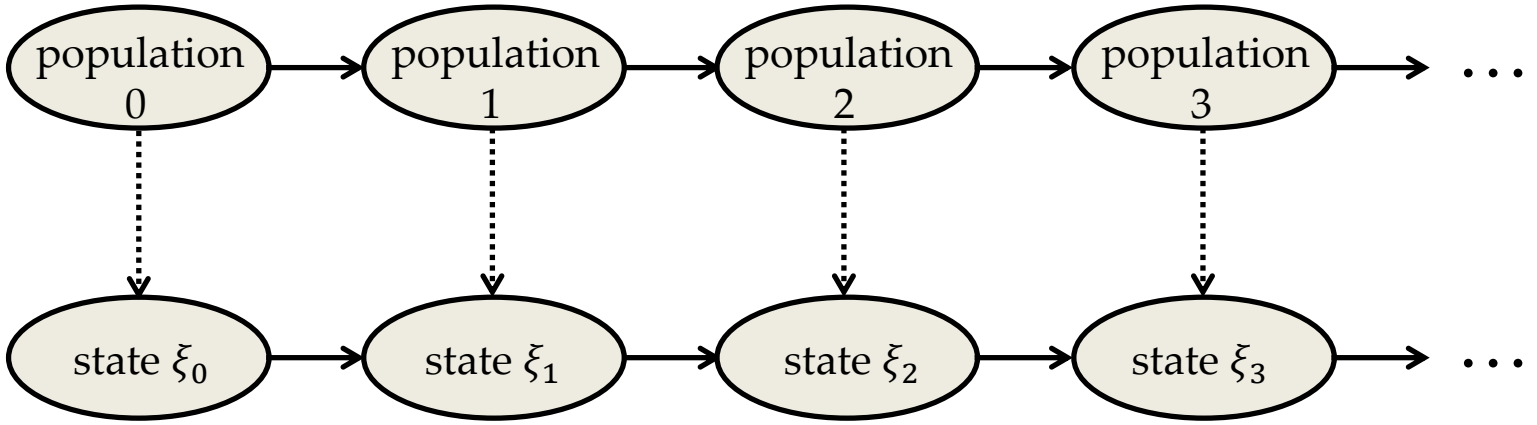
Theorem 1: $\xi \in \mathcal{X}$ modeling a MOEA solving a multi-objective problem, a well-defined function $h_{\boldsymbol{a},c}: \mathcal{X} \to \mathbb{N}_0$ and a Markov chain $\xi' \in \mathcal{Y} = \{0,1\}^r$ with $\mathcal{Y}^* = \{1^r\}$ such that $\forall x \notin \mathcal{X}^*, \quad \forall t \geq 0$,

$$\sum_{i\in[r]} \mathrm{P}(\min\{h(\xi_{t+1}), r\} = i | \xi_t = x) \mathrm{E}\left(\tau' \middle| \xi'_0 = 1^i 0^{r-i}\right)$$

$$\leq \sum_{y\in\mathcal{Y}} \mathrm{P}\left(\xi'_1 = y \middle| \xi'_0 = 1^{h(x)} 0^{r-h(x)}\right) \mathrm{E}(\tau' | \xi'_1 = y) + \delta,$$

$$\Rightarrow \mathrm{E}(\tau | \xi_0 = x_0) \leq \mathrm{E}(\tau' | \xi'_0 = 1^{\min\{h(x_0),r\}} 0^{r-\min\{h(x_0),r\}}) / (1-\delta)$$

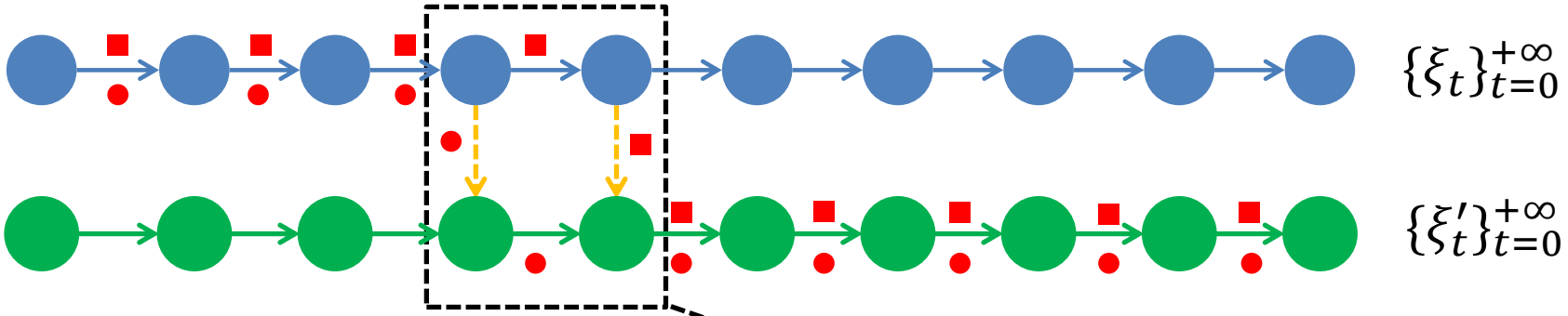
Main idea:



Switch analysis

Main idea [Bian, Qian and Tang, IJCAI'18]:

Given MOEA on the given problem



Reference chain

examine the different behaviors at each step

The expected running time of $\{\xi_t\}_{t=0}^{+\infty}$:

$$\mathrm{E}[\tau] \leq (\geq)\ \mathrm{E}[\tau'] + \sum_{t=0}^{+\infty} \rho_t$$

The expected running time of $\{\xi'_t\}_{t=0}^{+\infty}$, easy to analyze

Application

a simple MOEA which explains the common structure of MOEAs

GSEMO	Problem	Previous result	Our result
Bi-objective	LOTZ	$O(n^3)$ [Giel, CEC'03]	$\leq 6n^3$
	COCZ	$O(n^2 \log n)$ [Qian et al., AIJ'13]	$\leq 3n^2 \log n$
Many-objective	mCOCZ	$O(n^{m+1})$ [Laumanns et al., TEC'04]	$O(n^m)$ for $m > 4$, $O(n^3 \log n)$ for $m = 4$
Approximate analysis	WOMM	____	$1/n$-approximation: $O(n^2(\log_l n + \log_l(w_n/w_1)))$

gives the leading constants

is asymptotically tighter

Switch analysis is general and powerful!

Outline

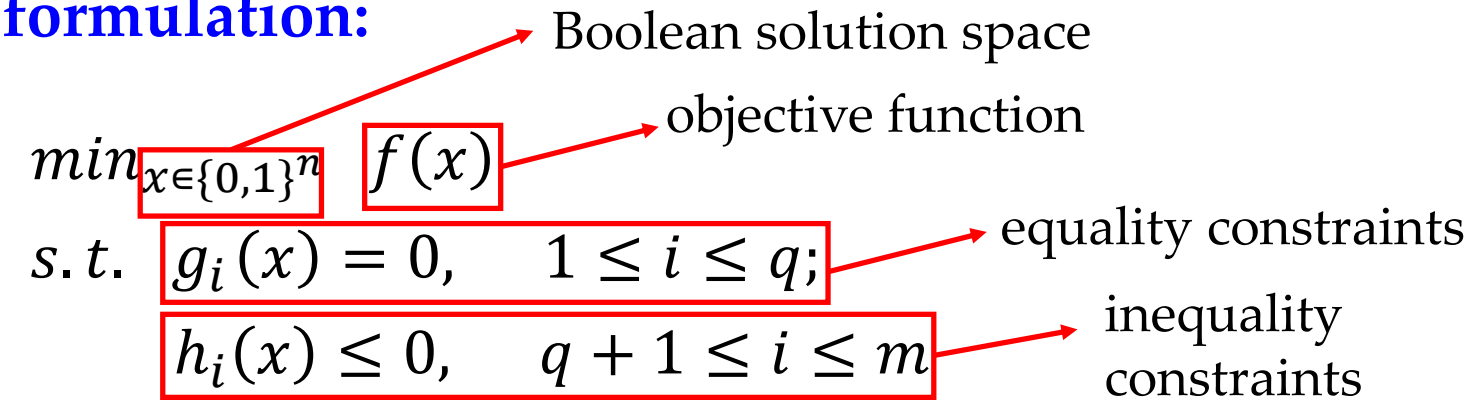
- Introduction
- Running time analysis approaches for MOEAs
- **Theoretical properties of MOEAs**
  - **Constrained optimization, noisy optimization**
- Multi-objective evolutionary learning algorithms
  - Selective ensemble, subset selection
- Conclusion

How about the performance of MOEAs for constrained optimization?

The optimization problems in machine learning often come with constraints

Constrained optimization

General formulation:



The goal: find a feasible solution minimizing the objective f

satisfies all constraints

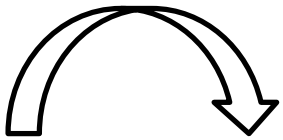
The penalty function method

Main idea [Hadj-Alouane & Bean, OR'97]

1. transform the original constrained optimization problem into an unconstrained optimization problem

constrained

$$\begin{aligned} min \quad & f(x) \\ s.t. \quad & g_i(x) = 0, \quad 1 \le i \le q; \\ & h_i(x) \le 0, \quad q+1 \le i \le m \end{aligned}$$



unconstrained

$$min \quad f(x) + \lambda \sum_{i=1}^{m} f_i(x)$$

constraint violation degree

$$f_i(x) = \begin{cases} |g_i(x)| & 1 \le i \le q \\ \max\{0, h_i(x)\} & q+1 \le i \le m \end{cases}$$

The penalty function method

Main idea [Hadj-Alouane & Bean, OR'97]

1. transform the original constrained optimization problem into an unconstrained optimization problem

$$min \quad f(x) + \lambda \sum_{i=1}^{m} f_i(x)$$

2. employ an unconstrained optimization algorithm to solve the transformed problem

Algorithm 1 (The Penalty Function Method) *Given a constrained optimization problem as in Eq. (1), it contains:*

1. *Let* $h(\boldsymbol{x}) = f(\boldsymbol{x}) + \lambda \sum_{i=1}^{m} f_i(\boldsymbol{x})$ *according to Eq. (2).*
2. $\boldsymbol{x}$ = *selected from* $\{0,1\}^n$ *uniformly at random.*
3. ***repeat*** *until the termination condition is met*
4. $\quad \boldsymbol{x}'$ = *flip each bit of* $\boldsymbol{x}$ *independently with prob.* $\frac{1}{n}$.
5. $\quad$ ***if*** $h(\boldsymbol{x}') \leq h(\boldsymbol{x})$
6. $\quad\quad \boldsymbol{x} = \boldsymbol{x}'$.
7. ***return*** $\boldsymbol{x}$

(1+1)-EA
[He & Yao, AIJ'01]

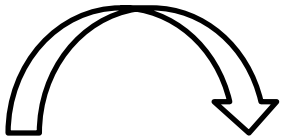
The Pareto optimization method

Main idea [Coello Coello, 2002; Cai & Wang, TEC'06]

1. transform the original constrained optimization problem into a bi-objective optimization problem

constrained

$$\begin{aligned} min \quad & f(x) \\ s.t. \quad & g_i(x) = 0, \quad 1 \le i \le q; \\ & h_i(x) \le 0, \quad q+1 \le i \le m \end{aligned}$$



bi-objective

$$min \quad (f(x), \sum_{i=1}^{m} f_i(x))$$

constraint violation degree

$$f_i(x) = \begin{cases} |g_i(x)| & 1 \le i \le q \\ \max\{0, h_i(x)\} & q+1 \le i \le m \end{cases}$$

The Pareto optimization method

Main idea [Coello Coello, 2002; Cai & Wang, TEC'06]

1. transform the original constrained optimization problem into a bi-objective optimization problem

$$min \quad (f(x),\ \sum_{i=1}^{m} f_i(x))$$

2. employ a multi-objective evolutionary algorithm to solve the transformed problem

Algorithm 2 (The Pareto Optimization Method) *Given a constrained optimization problem as in Eq. (1), it contains:*

1. *Let* $\boldsymbol{g}(\boldsymbol{x}) = (f(\boldsymbol{x}), \sum_{i=1}^{m} f_i(\boldsymbol{x}))$.
2. $\boldsymbol{x}$ = *selected from* $\{0,1\}^n$ *uniformly at random.*
3. $P = \{\boldsymbol{x}\}$.
4. ***repeat*** *until the termination condition is met*
5. $\quad \boldsymbol{x}$ = *selected from* P *uniformly at random.*
6. $\quad \boldsymbol{x}'$ = *flip each bit of* $\boldsymbol{x}$ *independently with prob.* $\frac{1}{n}$.
7. $\quad$ ***if*** $\nexists \boldsymbol{z} \in P$ *such that* $\boldsymbol{z} \succ_{\boldsymbol{g}} \boldsymbol{x}'$
8. $\quad\quad P = (P - \{\boldsymbol{z} \in P | \boldsymbol{x}' \succ_{\boldsymbol{g}} \boldsymbol{z}\}) \cup \{\boldsymbol{x}'\}$.
9. ***return*** $\boldsymbol{x} \in P$ *with* $\sum_{i=1}^{m} f_i(\boldsymbol{x}) = 0$

GSEMO
[Laumanns et al., TEC'04]

The Pareto optimization method

Main idea [Coello Coello, 2002; Cai & Wang, TEC'06]

1. transform the original constrained optimization problem into a bi-objective optimization problem

$$min \quad (f(x),\ \sum_{i=1}^{m} f_i(x))$$

2. employ a multi-objective evolutionary algorithm to solve the transformed problem

3. output the feasible solution from the generated non-dominated solution set

constraint violation degree = 0

Algorithm 2 (The Pareto Optimization Method) *Given a constrained optimization problem as in Eq. (1), it contains:*

1. *Let* $\boldsymbol{g}(\boldsymbol{x}) = (f(\boldsymbol{x}), \sum_{i=1}^{m} f_i(\boldsymbol{x}))$.
2. $\boldsymbol{x}$ = *selected from* $\{0,1\}^n$ *uniformly at random.*
3. $P = \{\boldsymbol{x}\}$.
4. ***repeat*** *until the termination condition is met*
5. $\quad \boldsymbol{x}$ = *selected from* P *uniformly at random.*
6. $\quad \boldsymbol{x}'$ = *flip each bit of* $\boldsymbol{x}$ *independently with prob.* $\frac{1}{n}$.
7. $\quad$ ***if*** $\nexists \boldsymbol{z} \in P$ *such that* $\boldsymbol{z} \succ_{\boldsymbol{g}} \boldsymbol{x}'$
8. $\quad\quad P = (P - \{\boldsymbol{z} \in P | \boldsymbol{x}' \succ_{\boldsymbol{g}} \boldsymbol{z}\}) \cup \{\boldsymbol{x}'\}$.
9. ***return*** $\boldsymbol{x} \in P$ *with* $\sum_{i=1}^{m} f_i(\boldsymbol{x}) = 0$

Problems

- Minimum matroid optimization (P-solvable) [Edmonds, MP'71]

  e.g., minimum spanning tree, maximum bipartite matching

Definition 1. Given a matroid (U, S), a rank function $r: 2^U \to \mathbb{N}$ and a weight function $w: U \to \mathbb{N}$, the problem is formulated as

$$min_{x \in \{0,1\}^n} \sum_{i=1}^{n} w_i x_i \quad s.t. \quad r(x) = r(U)$$

- Minimum cost coverage (NP-hard) [Wolsey, Combinatorica'82]

  e.g., minimum set cover, submodular set cover

Definition 2. Given a monotone submodular function $f: 2^U \to \mathbb{R}$, some value $q \leq f(U)$ and a weight function $w: U \to \mathbb{N}$, the problem is formulated as

$$min_{x \in \{0,1\}^n} \sum_{i=1}^{n} w_i x_i \quad s.t. \quad f(x) \geq q$$

Theoretical analysis

Penalty function vs. Pareto optimization

[Qian, Yu and Zhou, IJCAI'15]

- Minimum matroid optimization (P-solvable): obtaining an optimal solution

matroid rank, problem size, maximum weight

Penalty function: $\Omega(r^2 n(\log n + \log w_{max}))$

Pareto optimization: $O(rn(\log n + \log w_{max} + r))$

The running time reduces by a factor $\min\{\log n + \log w_{max}, r\}$

Theoretical analysis

- Minimum matroid optimization (P-solvable): obtaining an optimal solution

  Penalty function: $\Omega(r^2 n(\log n + \log w_{max}))$

  Pareto optimization: $O(rn(\log n + \log w_{max} + r))$

  The running time reduces by a factor $\min\{\log n + \log w_{max}, r\}$

- Minimum cost coverage (NP-hard): obtaining a H_q-approximate solution

  Penalty function: exponential w.r.t. $n, q, \log w_{max}$

  Pareto optimization: $O(qn(\log n + \log w_{max} + q))$ → polynomial

  The running time reduces exponentially

Theoretical analysis

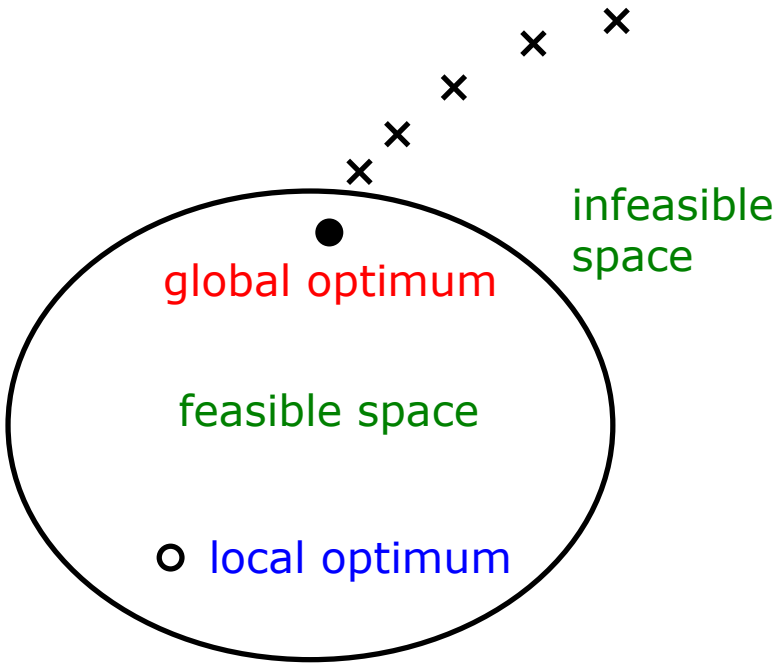
Findings from the analysis:

The penalty function method

- the penalty prefers feasible solutions
- get trapped in the local optimum, which is far from the global optimum

The Pareto optimization method

- the constraint violation objective allows infeasible solutions
- follow a short path from infeasible to feasible to find good solutions



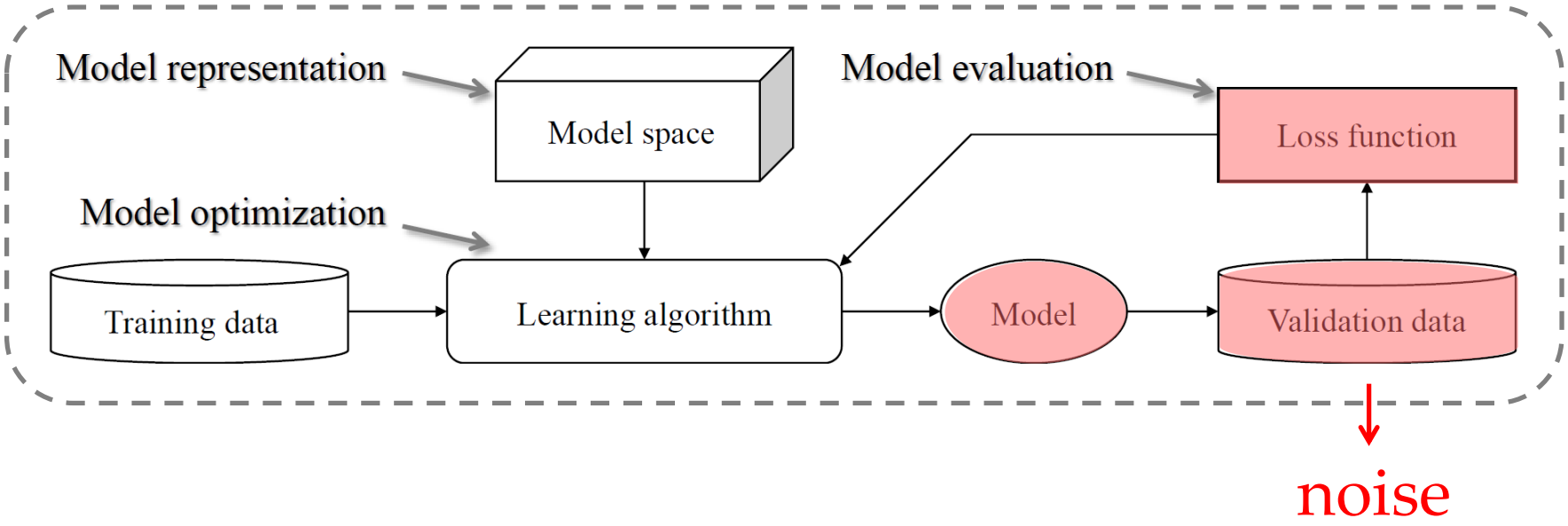
How about the performance for noisy optimization?

Previous theoretical analyses assume a clean environment, while optimization in machine learning often comes with noise

Noisy optimization

The objective evaluation is often disturbed by noise

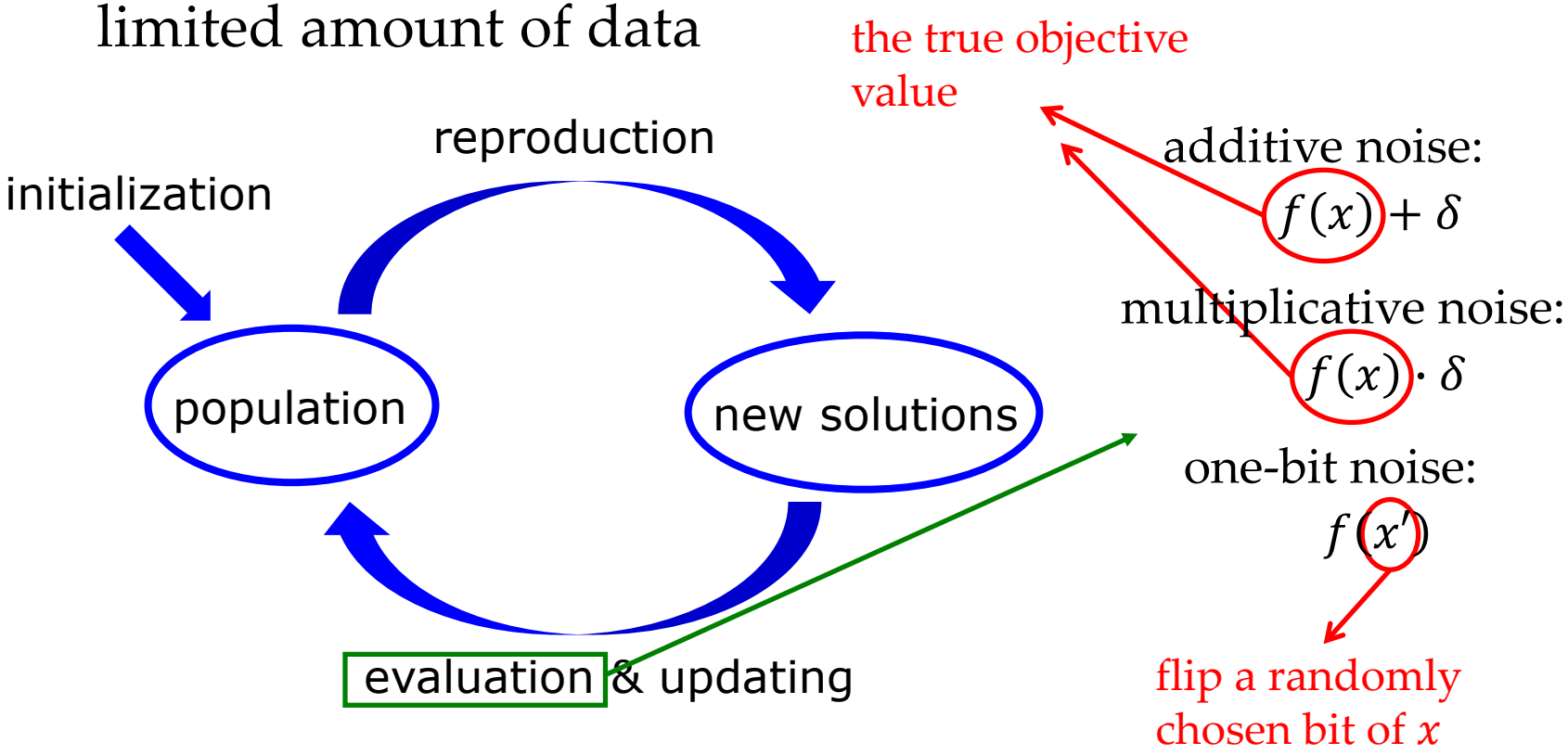
e.g., a prediction model is evaluated only on a limited amount of data



Noisy optimization

The objective evaluation is often disturbed by noise

e.g., a prediction model is evaluated only on a limited amount of data



The influence of noise

It was believed that noise makes evolutionary optimization harder

many noise handling strategies have been proposed
[Jin & Branke, TEC'05; Goh & Tan, TEC'07]

Some empirical observations have shown that noise can have a positive impact on the performance of local search
[Selman et al., AAAI'94; Hoos & Stutzle, JAR'00]

Can noise make evolutionary optimization easier?

Theoretical analysis

A sufficient condition: noise is helpful [Qian, Yu and Zhou, ECJ'18]

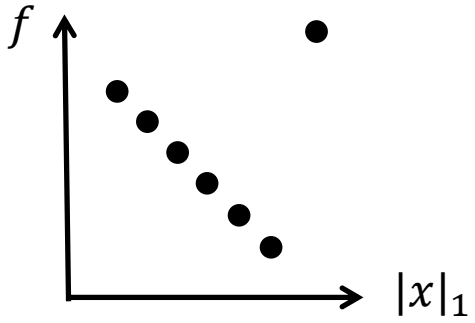
Theorem 1. *For an EA $\mathcal{A}$ optimizing a problem f, which can be modeled by a deceptive Markov chain, if*

$$\forall x \notin \mathcal{X}_0 : P_\xi^t(x, \mathcal{X}_0) = \sum\nolimits_{x' \cap \mathcal{S}^* \neq \emptyset} P_{var}(x, x'), \tag{6}$$

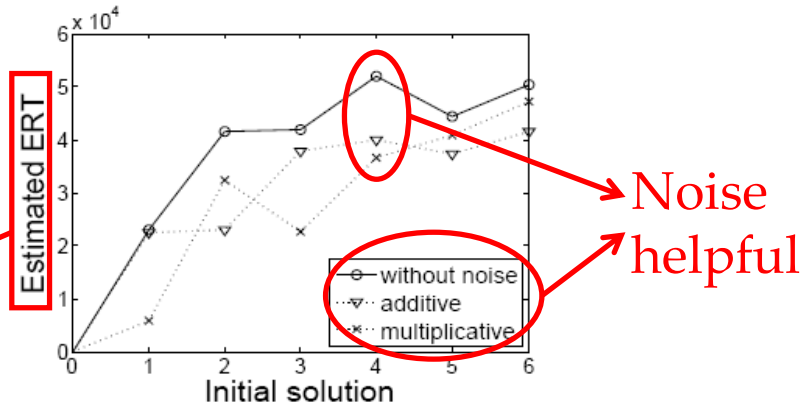
then noise makes f easier for $\mathcal{A}$.

Intuitively, if an EA searches along the deceptive direction, noise can add some randomness to make the EA run along the right direction

Example: (1+n)-EA on the Trap problem



Running time



The influence of noise

Hypothesis: the negative influence of noise decreases as the problem hardness increases

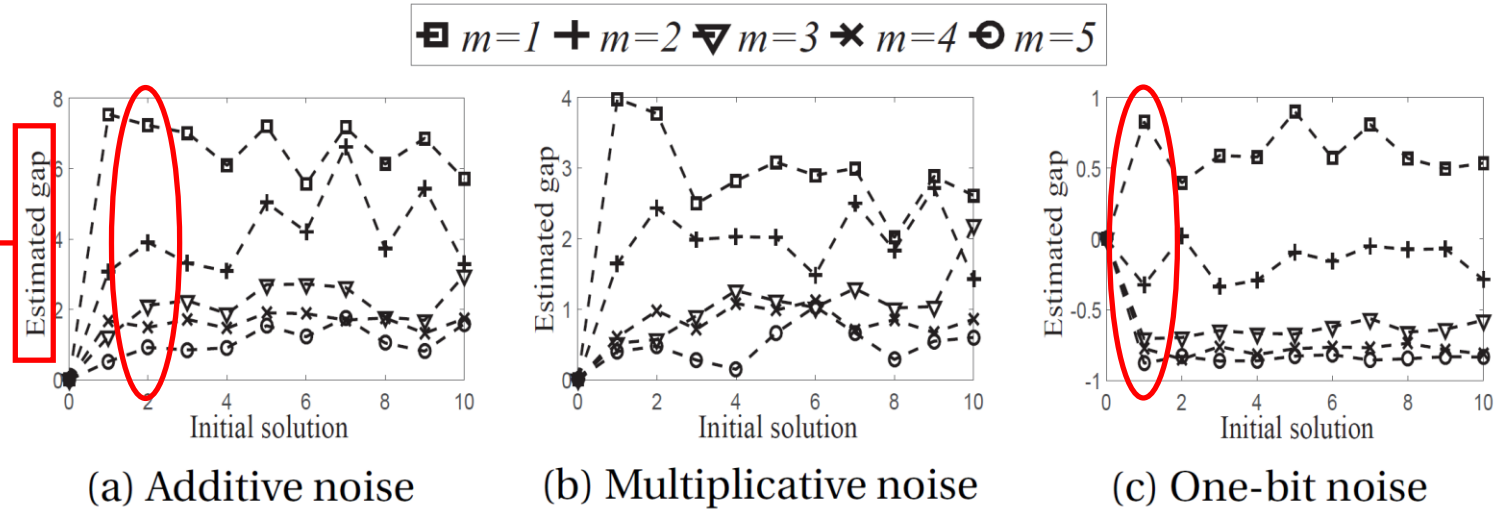
Empirical verification: (1+1)-EA on the $\text{Jump}_{m,n}$ problem with $\Theta(n^m + n\log n)$ running time [Droste et al., TCS'02]

Larger m, harder the problem

Running time with noise

$$\frac{\mathrm{E}[\tau] - \mathrm{E}[\tau']}{\mathrm{E}[\tau']}$$

Running time without noise



Noise may be helpful when the problem is quite hard

Noise handling strategies

Noise is harmful in most cases

Two commonly used noise handling strategies:

- Re-evaluation [Arnold & Beyer, TEC'02; Jin & Branke, TEC'05]
  - every time we access the fitness of a solution by evaluation

smooth noise

- Threshold selection [Markon et al., CEC'01; Bartz-Beielstein & Markon, CEC'02]
  - an offspring solution is accepted only if its fitness is larger than that of the parent solution by at least a threshold τ

reduce the risk of accepting a bad solution due to noise

Theoretical analysis

the range of noise level such that the running time is polynomial

Example:

(1+1)-EA

OneMax

one-bit noise

Noise handling strategies	PNT
single-evaluation	$[0, 1 - \frac{1}{\Theta(poly(n))}]$
single-evaluation & $\tau > 0$	$[0, 0]$
re-evaluation	$[0, \Theta(\frac{\log n}{n})]$ (Droste, 2004)
re-evaluation & $\tau = 1$	$[0, 1]$
re-evaluation & $\tau = 2$	$[\frac{1}{\Theta(poly(n))}, 1 - \frac{1}{\Theta(poly(n))}]$
re-evaluation & $\tau > 2$	$\emptyset$

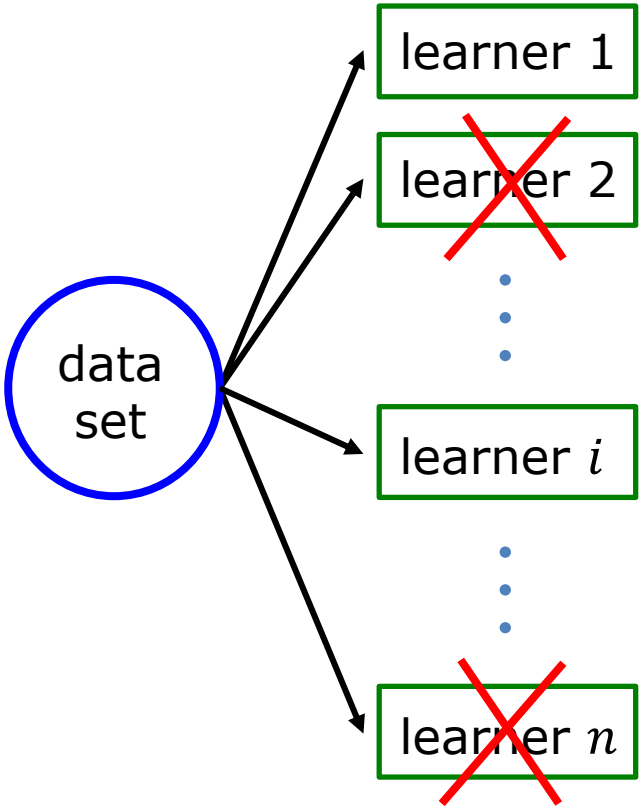
combining re-evaluation with proper threshold selection is better

Outline

- Introduction
- Running time analysis approaches for MOEAs
- Theoretical properties of MOEAs
  - Constrained optimization, noisy optimization
- **Multi-objective evolutionary learning algorithms**
  - **Selective ensemble, subset selection**
- Conclusion

Back to selective ensemble

Selective ensemble [Zhou, 2012]



Two goals:

- maximize the generalization performance
- minimize the number of selected learners

The PEP approach

PEP (Pareto Ensemble Pruning) [Qian, Yu and Zhou, AAAI'15]

Main idea:

optimize the two goals of selective ensemble simultaneously by MOEAs

Algorithm 1 (PEP). *Given a set of trained classifiers* $H = \{h_i\}_{i=1}^{n}$, *an objective* $f: 2^H \to \mathbb{R}$ *and an evaluation criterion eval, it contains:*

1. *Let* $\boldsymbol{g}(s) = (f(H_s), |s|)$ *be the bi-objective.*
2. *Let* s = *randomly selected from* $\{0,1\}^n$ *and* $P = \{s\}$.
3. ***Repeat***
4. *Select* $s \in P$ *uniformly at random.*
5. *Generate* s' *by flipping each bit of* s *with prob.* $\frac{1}{n}$.
6. ***if*** $\nexists z \in P$ *such that* $z \succ_{\boldsymbol{g}} s'$
7. $P = (P - \{z \in P \mid s' \succeq_{\boldsymbol{g}} z\}) \cup \{s'\}$.
8. $Q = VDS(f, s')$.
9. ***for*** $q \in Q$
10. ***if*** $\nexists z \in P$ *such that* $z \succ_{\boldsymbol{g}} q$
11. $P = (P - \{z \in P \mid q \succeq_{\boldsymbol{g}} z\}) \cup \{q\}$.
12. ***Output*** $\arg\min_{s \in P} eval(s)$.

Initialization: randomly generate a solution, put it into the population P

Reproduction: pick a solution randomly from P, and mutate it to make a new one

Evaluation & Updating: if the new solution is not dominated, put it and its good neighbors into P

Output: select a final solution

Previous approaches

- Ordering-based methods (OEP)

  Main idea: give an order of base classifiers according to some criterion, and select the front classifiers

  - error minimization [Margineantu & Dietterich, ICML'97]
  - diversity-like criterion maximization [Martínez-Munõz et al., TPAMI'09]
  - combined criterion [Li et al., ECML'12]

- Single-objective optimization-based methods (SEP)

  Main idea: formulate selective ensemble as a single-objective optimization problem, and employ some optimization technique

  - semi-definite programming [Zhang et al., JMLR'06]
  - quadratic programming [Li & Zhou, MCS'09]
  - genetic algorithms [Zhou et al., AIJ'02]

Theoretical analysis

PEP is at least as good as ordering-based methods

Theorem 1. For any objective and any size, PEP within $O(n^4 \log n)$ expected optimization time can find a solution weakly dominating that generated by OEP at the fixed size.

PEP can be better than ordering-based methods

Theorem 2. In Situation 1, OEP using Eq.1 finds a solution with objective vector $(\geq 0, \geq 3)$ where the two equalities never hold simultaneously, while PEP finds a solution with objective vector $(0, 3)$ in $O(n^4 \log n)$ expected time.

PEP/OEP can be better than single-objective optimization-based methods

Theorem 3. In Situation 2, OEP using Eq.1 finds the optimal solution in $O(n^2)$ optimization time, while the time of SEP is at least $2^{\Omega(n)}$ with probability $1 - 2^{-\Omega(n)}$.

Experimental results - test error

Pruning bagging base learners with size 100

baseline methods | ordering-based methods | optimization-based methods

Test Error

Data set	PEP	Bagging	BI	RE	Kappa	CP	MD	DREP	EA
australian	.144±.020	**.143±.017**	.152±.023•	.144±.020	**.143±.021**	.145±.022	.148±.022	.144±.019	**.143±.020**
breast-cancer	**.275±.041**	.279±.037	.298±.044•	.277±.031	.287±.037	.282±.043	.295±.044•	**.275±.036**	**.275±.032**
disorders	**.304±.039**	.327±.047•	.365±.047•	.320±.044•	.326±.042•	.306±.039	.337±.035•	.316±.045	.317±.046•
heart-statlog	.197±.037								
house-votes	.045±.019								
ionosphere	.088±.021								
kr-vs-kp	**.010±.003**								
letter-ah	.013±.005	.021±.006•	.023±.008•	.015±.006•	**.012±.006**	.015±.006	.017±.007•	.014±.005	.017±.006•
letter-br	**.046±.008**	.059±.013•	.078±.012•	.048±.012	.048±.014	.048±.012	.057±.014•	.048±.009	.053±.011•
letter-oq	.043±.009								
optdigits	**.035±.006**								
satimage-12v57	**.028±.004**								
satimage-2v5	**.021±.007**								
sick	**.015±.003**	.018±.004•	.018±.004•	.016±.003	.017±.003•	.016±.003•	.017±.003•	.016±.003	.017±.004•
sonar	**.248±.056**	.266±.052	.310±.051•	.267±.053•	.249±.059	.250±.048	.268±.055•	.257±.056	.251±.041
spambase	**.065±.006**	.068±.007•	.093±.008•	.066±.006	.066±.006	.066±.006	.068±.007•	**.065±.006**	.066±.006
tic-tac-toe	.131±.027	.164±.028•	.212±.028•	.135±.026	.132±.023	.132±.026	.145±.022•	**.129±.026**	.138±.020
vehicle-bo-vs	**.224±.023**	.228±.026	.257±.025•	.226±.022	.233±.024•	.234±.024•	.244±.024•	.234±.026•	.230±.024
vehicle-b-v	**.018±.011**	.027±.014•	.024±.013•	.020±.011	.019±.012	.020±.011	.021±.011•	.019±.013	.026±.013•
vote	.044±.018	.047±.018	.046±.016	.044±.017	**.041±.016**	.043±.016	.045±.014	.043±.019	.045±.015
count of the best	**12**	2	0	2	7	1	0	5	5
PEP: count of direct win		**17**	**20**	**15.5**	12.5	**17**	**20**	12.5	**15.5**

PEP achieves the smallest error on 60% (12/20) of the data sets, while other methods are less than 35% (7/20)

PEP is better than any other method on more than 60% (12.5/20) data sets

PEP is never significantly worse

Experimental results - ensemble size

ordering-based methods ← optimization-based methods

Ensemble Size

Data set	PEP	RE	Kappa	CP	MD	DREP	EA
australian	10.6±4.2	12.5±6.0	14.7±12.6	11.0±9.7	**8.5±14.8**	11.7±4.7	41.9±6.7•
breast-cancer	8.4±3.5	8.7±3.6	26.1±21.7•	8.8±12.3	**7.8±15.2**	9.2±3.7	44.6±6.6•
disorders	14.7±4.2						
heart-statlog	**9.3±2.3**						
house-votes	**2.9±1.7**						
ionosphere	**5.2±2.2**						
kr-vs-kp	**4.2±1.8**						
letter-ah	**5.0±1.9**						
letter-br	**10.9±2.6**	15.1±7.3•	13.8±6.7•	12.9±6.8	23.2±17.6•	11.3±3.5	38.3±7.8•
letter-oq	**12.0±3.7**						
optdigits	22.7±3.1						
satimage-12v57	**17.1±5.0**						
satimage-2v5	**5.7±1.7**						
sick	**6.9±2.8**						
sonar	11.4±4.2	**11.0±4.1**	20.6±9.3•	13.9±7.1	20.6±20.7•	14.4±5.9•	43.1±6.4•
spambase	17.5±4.5	18.5±5.0	20.0±8.1	19.0±9.9	28.8±17.0•	**16.7±4.6**	39.7±6.4•
tic-tac-toe	14.5±3.8	16.1±5.4	17.4±6.5	15.4±6.3	28.0±22.6•	**13.6±3.4**	39.8±8.2•
vehicle-bo-vs	16.5±4.5	15.7±5.7	16.5±8.2	**11.2±5.7**○	21.6±20.4	3.2±5.0○	41.9±5.6•
vehicle-b-v	**2.8±1.1**	3.4±2.1	4.5±1.6•	5.3±7.4	**2.8±3.8**	4.0±3.9	48.0±5.6•
vote	**2.7±1.1**	3.2±2.7	5.1±2.6•	5.4±5.2•	6.0±9.8	3.9±2.5•	47.8±6.1•
count of the best	12	2	0	2	3	3	0
PEP: count of direct win		**17**	**19.5**	**18**	**17.5**	**16**	**20**

PEP achieves the smallest size on 60% (12/20) of the data sets, while other methods are less than 15% (3/20)

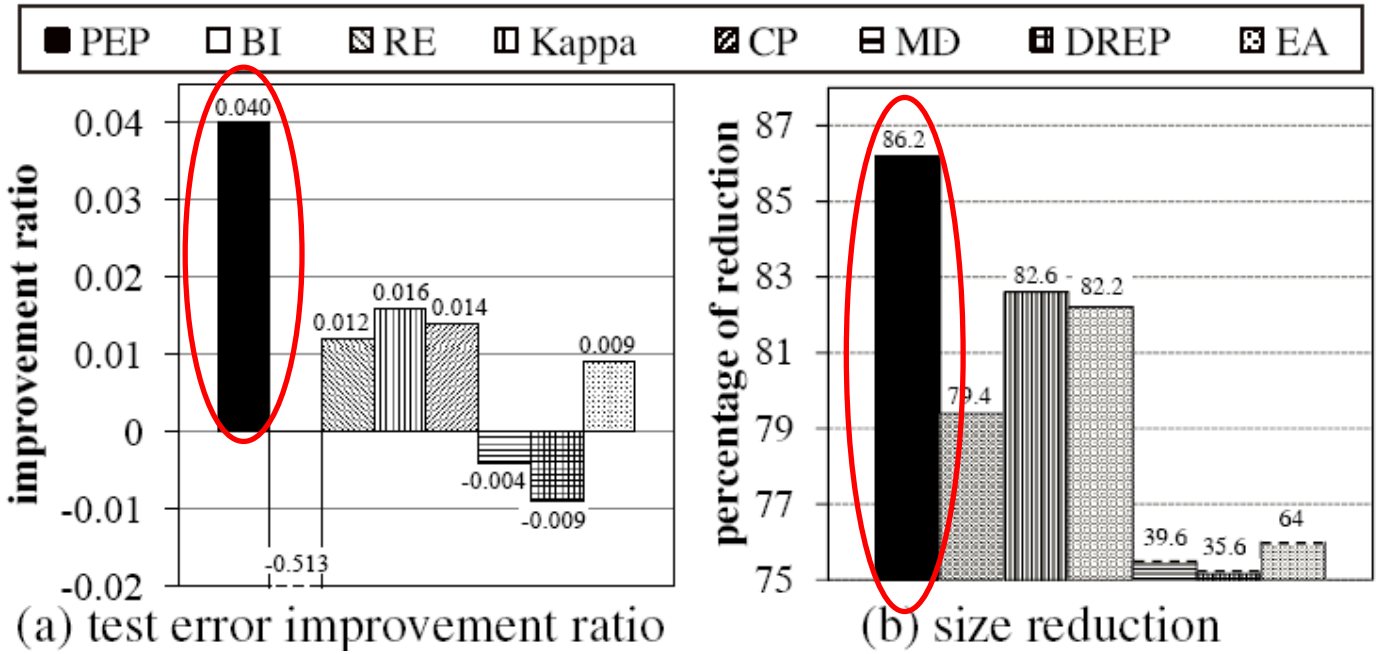
PEP is better than any other method on more than 80% (16/20) data sets

PEP is never significantly worse, except two losses on vehicle-bo-vs

Application

Mobile Human Activity Recognition: identify the actions carried out by a person according to the information gathered by smartphones

On a public data set [Anguita et al., IWAAL'12]: 6 activities



multiclass SVM: 89.3%
[Anguita et al., IWAAL'12]

PEP: 90.4%

3 times more than the runner up

save more than 20% storage and testing time than the runner up

We developed a Pareto optimization method for

Selective ensemble

- minimize the number of selected learners
- optimize the generalization performance

Subset selection

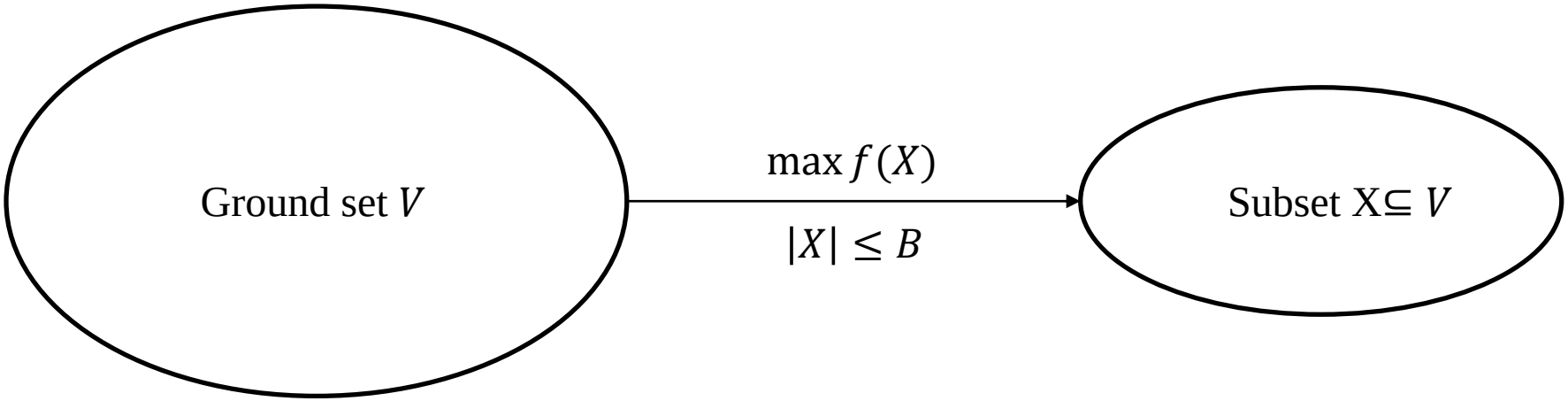
- minimize the number of selected items
- optimize a given objective function

Subset selection

Subset selection is to select a subset of size B from a total set of n items for optimizing some objective function

Formally stated: given all items $V = \{v_1, \ldots, v_n\}$, an objective function $f: 2^V \rightarrow \mathrm{R}$ and a budget B, to find a subset $X \subseteq V$ such that

$$max_{X \subseteq V} \quad f(X) \quad s.t. \quad |X| \leq B.$$



Sparse regression

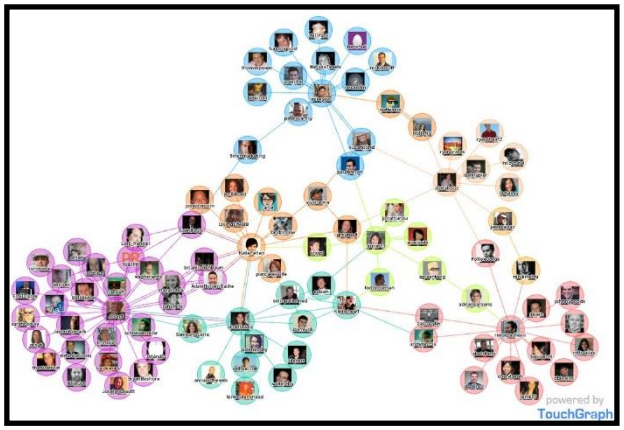
Sparse regression [Tropp, TIT'04] : find a sparse approximation solution to the linear regression problem

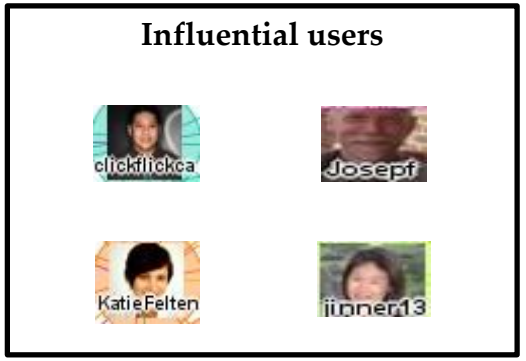
	Corr.	Dis.	LR	···	···	AIC.	BIC	RF.
x1	0.28	0.46	1	···	···	0.22	0.63	1
x2	0.31	0.59	0.64	···	···	0.58	0.56	1
x3	0.11	0.02	0.53	···	···	0.43	0.01	1
x4	0.1	0.1	0.64	···	···	0.73	0.92	1
x5	0.02	0.15	0.33	···	···	0.56	0.36	0.78
x6	0.36	0.02	0.01	···	···	0.32	0.02	0.22
x7	0.2	0.2	0.21	···	···	0.21	0.02	0.11
x8	0.1	0.03	0.32	···	···	0.33	0.51	0.44
x9	0.32	0.1	0.2	···	···	0.06	0.66	0
x10	0.24	0	0.02	···	···	0.6	0.03	0.33
x11	0.12	0.45	0.44	···	···	0.64	0.45	1
x12	0.36	0.58	0.12	···	···	0.73	0.58	0.67
x13	0.2	0.02	0.24	···	···	0.34	0.02	0.89
x14	0.24	0.92	0.33	···	···	0.24	0.93	0.56

	Corr.	Dis.	LR	···	···	AIC.	BIC	RF.
x1	0.28	0.46	1	···	···	0.22	0.63	1
x2	0.31	0.59	0.64	···	···	0.58	0.56	1
x3	0.11	0.02	0.53	···	···	0.43	0.01	1
x4	0.1	0.1	0.64	···	···	0.73	0.92	1
x5	0.02	0.15	0.33	···	···	0.56	0.36	0.78
x6	0.36	0.02	0.01	···	···	0.32	0.02	0.22
x7	0.2	0.2	0.21	···	···	0.21	0.02	0.11
x8	0.1	0.03	0.32	···	···	0.33	0.51	0.44
x9	0.32	0.1	0.2	···	···	0.06	0.66	0
x10	0.24	0	0.02	···	···	0.6	0.03	0.33
x11	0.12	0.45	0.44	···	···	0.64	0.45	1
x12	0.36	0.58	0.12	···	···	0.73	0.58	0.67
x13	0.2	0.02	0.24	···	···	0.34	0.02	0.89
x14	0.24	0.92	0.33	···	···	0.24	0.93	0.56

Influence maximization

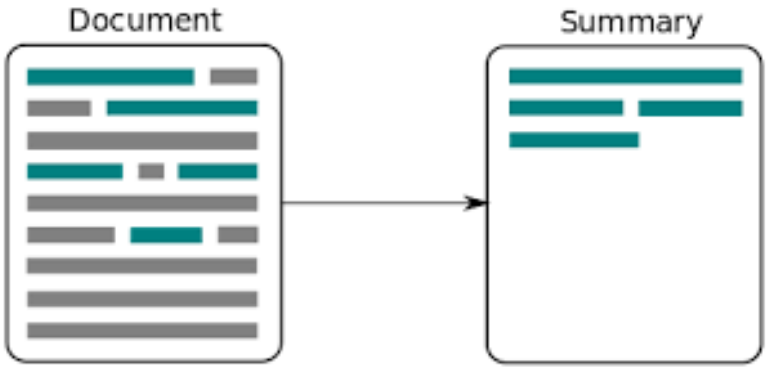
Influence maximization [Kempe et al., KDD'03] : select a subset of users from a social network to maximize its influence spread

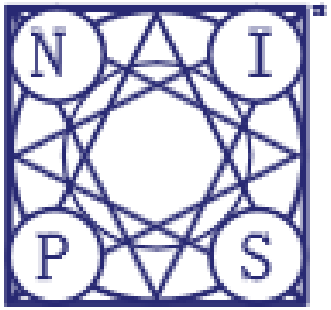


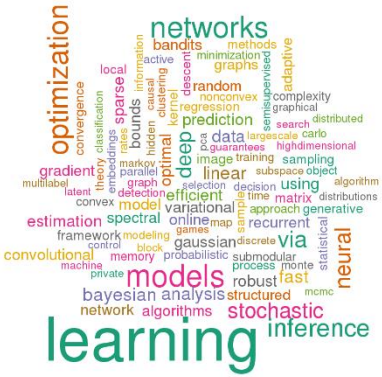


Document summarization

Document summarization [Lin & Bilmes, ACL'11]: select a few sentences to best summarize the documents

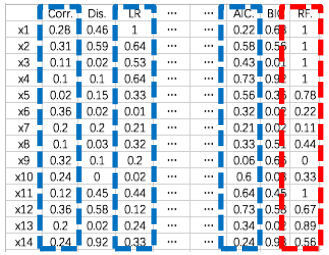






Subset selection

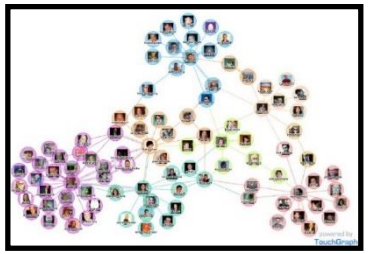
Sparse regression



Machine learning

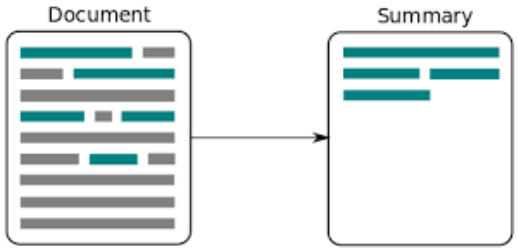
Data mining

Influence maximization



Subset selection

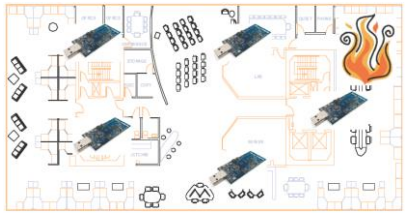
Document summarization

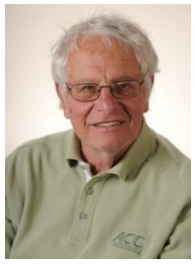


Natural language processing

Information retrieval

Sensor placement





George Nemhauser

John Von Neumann Theory Prize

[Mathematical Programming 1978]

f : monotone and submodular

The greedy algorithm:

$(1 - 1/e)$-approximation

Best Paper:

[Das & Kempe, ICML'11]

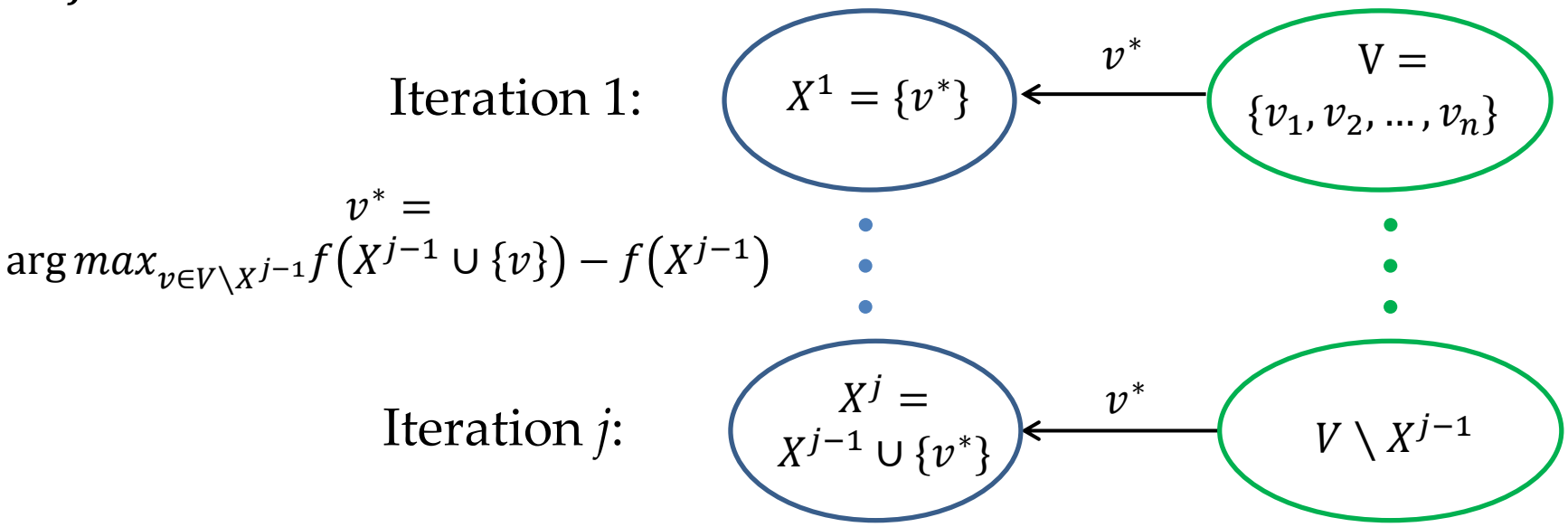
[Iyer, et al., ICML'13]

[Iyer & Bilmes, NIPS'13]

The greedy algorithm

Subset selection: $max_{X \subseteq V} \quad f(X) \quad s.t. \quad |X| \leq B$

Process: iteratively select one item that makes the increment on f maximized



The optimal approximation guarantee [Nemhauser & Wolsey, MOR'78] :

$1 - 1/e \approx 0.632$ by the greedy algorithm

The POSS approach

The POSS approach [Qian, Yu and Zhou, NIPS'15]

$$max_{x\in\{0,1\}^n}\ f(x) \quad s.t. \quad |x| \le B \qquad \text{original}$$

Transformation: ⇩

$$min_{x\in\{0,1\}^n}\ (-f(x), |x|) \qquad \text{bi-objective}$$

Algorithm 1 POSS

Input: all variables $V = \{X_1, \ldots, X_n\}$, a given objective f and an integer parameter $k \in [1, n]$
Parameter: the number of iterations T
Output: a subset of V with at most k variables
Process:

1: Let $s = \{0\}^n$ and $P = \{s\}$.
2: Let $t = 0$.
3: **while** $t < T$ **do**
4: Select s from P uniformly at random.
5: Generate s' by flipping each bit of s with prob. $\frac{1}{n}$.
6: Evaluate $f_1(s')$ and $f_2(s')$.
7: **if** $\nexists z \in P$ such that $z \prec s'$ **then**
8: $Q = \{z \in P \mid s' \preceq z\}$.
9: $P = (P \setminus Q) \cup \{s'\}$.
10: **end if**
11: $t = t + 1$.
12: **end while**
13: **return** $\arg\min_{s\in P, |s|\le k} f_1(s)$

Initialization: put the special solution $\{0\}^n$ into the population P

Reproduction: pick a solution x randomly from P, and flip each bit of x with prob. $1/n$ to generate a new solution

Evaluation & Updating: if the new solution is not dominated, put it into P and weed out bad solutions

Output: select the best feasible solution

Theoretical analysis

POSS can achieve the same general approximation guarantee as the greedy algorithm

Theorem 1. For subset selection with monotone objective functions, POSS using $E[T] \leq 2eB^2n$ finds a solution x with $|x| \leq B$ and $f(x) \geq (1 - e^{-\gamma}) \cdot OPT$.

the expected number of iterations

the best known polynomial-time approximation ratio, previously obtained by the greedy algorithm [Das & Kempe, ICML'11]

POSS can do better than the greedy algorithm in cases

[Das & Kempe, STOC'08]

Theorem 2. For the Exponential Decay subclass of sparse regression, POSS using $E[T] = O(B^2(n - B)n \log n)$ finds an optimal solution, while the greedy algorithm cannot.

Sparse regression

Sparse regression [Tropp, TIT'04]: find a sparse approximation solution to the linear regression problem

Formally stated: given all observation variables $V = \{v_1, \dots, v_n\}$, a predictor variable z and a budget B, to find a subset $X \subseteq V$ such that

$$max_{X \subseteq V} \quad R^2_{z,X} = \frac{\mathrm{Var}(z) - \mathrm{MSE}_{z,X}}{\mathrm{Var}(z)} \quad s.t. \quad |X| \leq B.$$

	Corr.	Dis.	LR	···	···	AIC.	BIC	RF.
x1	0.28	0.46	1	···	···	0.22	0.63	1
x2	0.31	0.59	0.64	···	···	0.58	0.56	1
x3	0.11	0.02	0.53	···	···	0.43	0.01	1
x4	0.1	0.1	0.64	···	···	0.73	0.92	1
x5	0.02	0.15	0.33	···	···	0.56	0.36	0.78
x6	0.36	0.02	0.01	···	···	0.32	0.02	0.22
x7	0.2	0.2	0.21	···	···	0.21	0.02	0.11
x8	0.1	0.03	0.32	···	···	0.33	0.51	0.44
x9	0.32	0.1	0.2	···	···	0.06	0.66	0
x10	0.24	0	0.02	···	···	0.6	0.03	0.33
x11	0.12	0.45	0.44	···	···	0.64	0.45	1
x12	0.36	0.58	0.12	···	···	0.73	0.58	0.67
x13	0.2	0.02	0.24	···	···	0.34	0.02	0.89
x14	0.24	0.92	0.33	···	···	0.24	0.93	0.56

	Corr.	Dis.	LR	···	···	AIC.	BIC	RF.
x1	0.28	0.46	1	···	···	0.22	0.63	1
x2	0.31	0.59	0.64	···	···	0.58	0.56	1
x3	0.11	0.02	0.53	···	···	0.43	0.01	1
x4	0.1	0.1	0.64	···	···	0.73	0.92	1
x5	0.02	0.15	0.33	···	···	0.56	0.36	0.78
x6	0.36	0.02	0.01	···	···	0.32	0.02	0.22
x7	0.2	0.2	0.21	···	···	0.21	0.02	0.11
x8	0.1	0.03	0.32	···	···	0.33	0.51	0.44
x9	0.32	0.1	0.2	···	···	0.06	0.66	0
x10	0.24	0	0.02	···	···	0.6	0.03	0.33
x11	0.12	0.45	0.44	···	···	0.64	0.45	1
x12	0.36	0.58	0.12	···	···	0.73	0.58	0.67
x13	0.2	0.02	0.24	···	···	0.34	0.02	0.89
x14	0.24	0.92	0.33	···	···	0.24	0.93	0.56

Experimental results - R^2 values

the size constraint: $B = 8$ **the number of iterations of POSS: $2eB^2n$**

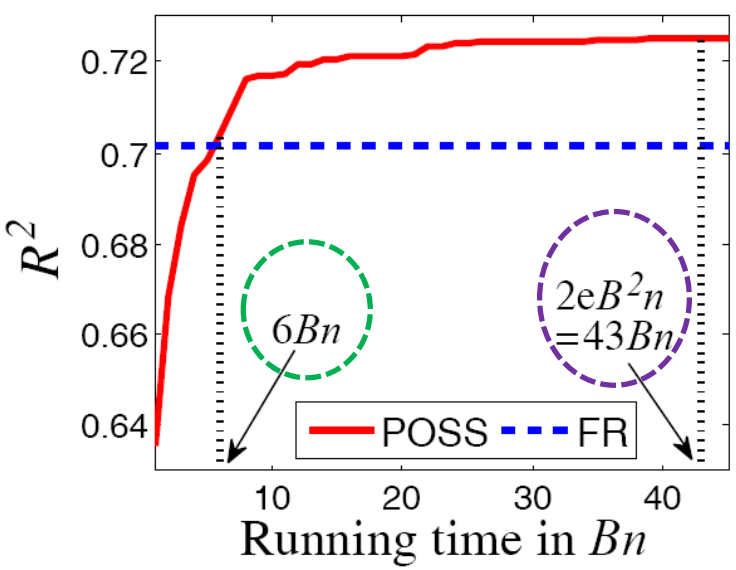
exhaustive search | greedy algorithms | relaxation methods

Data set	OPT	POSS	FR	FoBa	OMP	RFE	MCP
housing	.7437±.0297	.7437±.0297	.7429±.0300•	.7423±.0301•	.7415±.0300•	.7388±.0304•	.7354±.0297•
eunite2001	.8484±.0132	.8482±.0132	.8348±.0143•	.8442±.0144•	.8349±.0150•	.8424±.0153•	.8320±.0150•
svmguide3	.2705±.0255	.2701±.0257	.2615±.0260•	.2601±.0279•	.2557±.0270•	.2136±.0325•	.2397±.0237•
ionosphere	.5995±.0326	.5990±.0329	.5920±.0352•	.5929±.0346•	.5921±.0353•	.5832±.0415•	.5740±.0348•
sonar	–	.5365±.0410	.5171±.0440•	.5138±.0432•	.5112±.0425•	.4321±.0636•	.4496±.0482•
triazines	–	.4301±.0603	.4150±.0592•	.4107±.0600•	.4073±.0591•	.3615±.0712•	.3793±.0584•
coil2000	–	.0627±.0076	.0624±.0076•	.0619±.0075•	.0619±.0075•	.0363±.0141•	.0570±.0075•
mushrooms	–	.9912±.0020	.9909±.0021•	.9909±.0022•	.9909±.0022•	.6813±.1294•	.8652±.0474•
clean1	–	.4368±.0300	.4169±.0299•	.4145±.0309•	.4132±.0315•	.1596±.0562•	.3563±.0364•
w5a	–	.3376±.0267	.3319±.0247•	.3341±.0258•	.3313±.0246•	.3342±.0276•	.2694±.0385•
gisette	–	.7265±.0098	.7001±.0116•	.6747±.0145•	.6731±.0134•	.5360±.0318•	.5709±.0123•
farm-ads	–	.4217±.0100	.4196±.0101•	.4170±.0113•	.4170±.0113•	–	.3771±.0110•
POSS: win/tie/loss		–	12/0/0	12/0/0	12/0/0	11/0/0	12/0/0

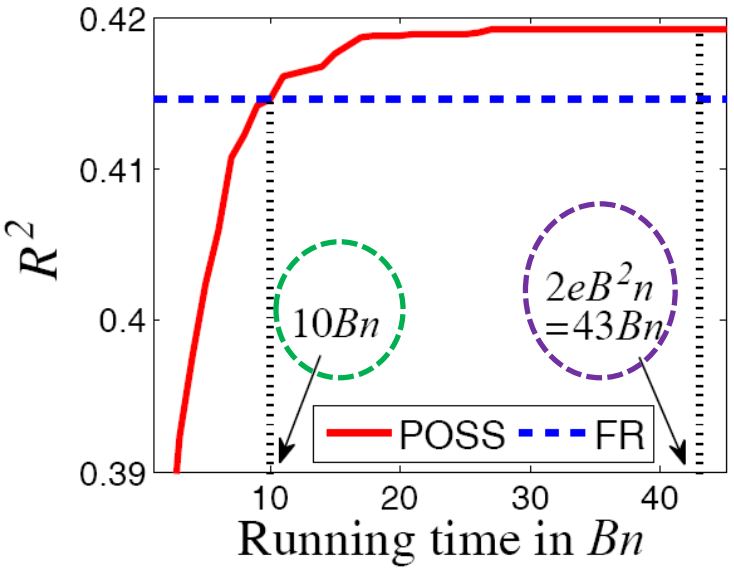
POSS is significantly better than all the compared methods on all data sets

Experimental results – running time

OPT: n^B/B^B greedy methods (FR): Bn POSS: $2eB^2n$



(a) on *gisette*



(b) on *farm-ads*

theoretical running time

POSS can be much more efficient in practice than in theoretical analysis

POSS vs. Greedy algorithm

Greedy algorithm:

- Generate a new solution by adding a single item (single-bit forward search: $0 \rightarrow 1$)
- Maintain only one solution

POSS:

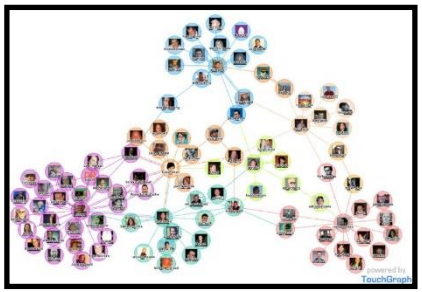
- Generate a new solution by flipping each bit of a solution with prob. $1/n$ (single-bit forward search, backward search, multi-bit search)
- Maintain several non-dominated solutions due to bi-objective optimization

POSS may have a better ability of avoiding local optima!

Noise

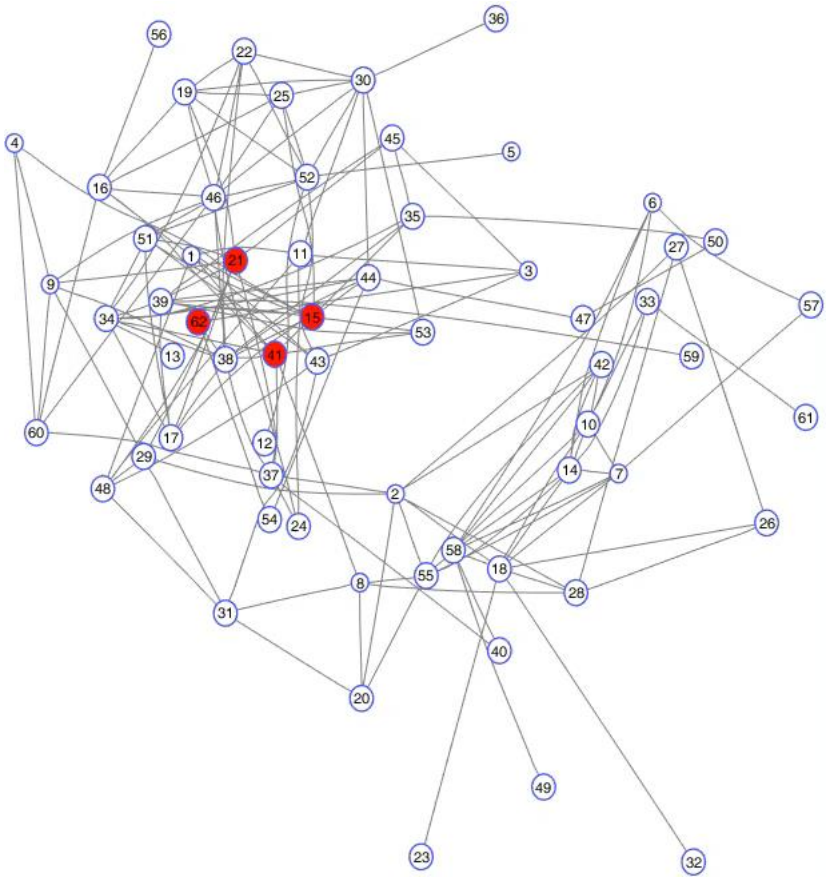
Previous analyses often assume that the exact value of the objective function can be accessed

However, in many applications of subset selection, only a noisy value of the objective function can be obtained



Influence maximization

The objective function: the expected number of nodes activated by propagating from X



1st diffusion: 15

2nd diffusion: 16

•
•
•

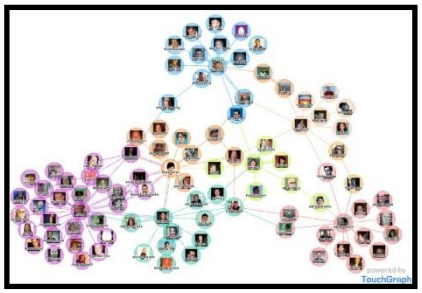
To achieve an accurate estimation, 10,000 independent diffusion processes are required

[Kempe et al., KDD'03]

Noise

Previous analyses often assume that the exact value of the objective function can be accessed

However, in many applications of subset selection, only a noisy value of the objective function can be obtained



Influence maximization

The objective function: the expected number of nodes activated by propagating from X

noise

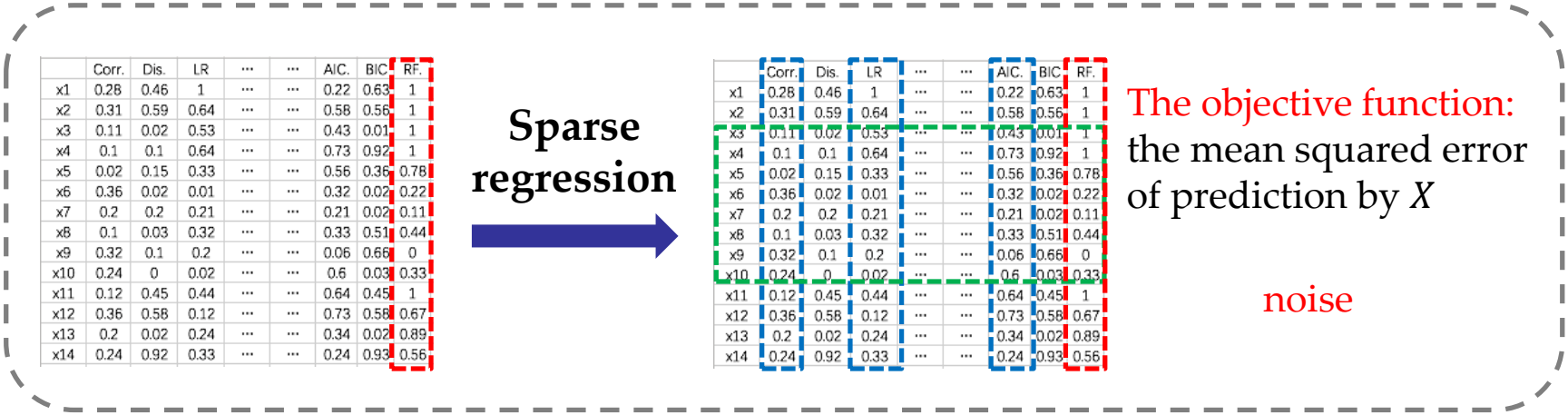
The average number of active nodes of independent diffusion processes

[Kempe et al., KDD'03]

Noise

Previous analyses often assume that the exact value of the objective function can be accessed

However, in many applications of subset selection, only a noisy value of the objective function can be obtained



	Corr.	Dis.	LR	···	···	AIC.	BIC	RF.
x1	0.28	0.46	1	···	···	0.22	0.63	1
x2	0.31	0.59	0.64	···	···	0.58	0.56	1
x3	0.11	0.02	0.53	···	···	0.43	0.01	1
x4	0.1	0.1	0.64	···	···	0.73	0.92	1
x5	0.02	0.15	0.33	···	···	0.56	0.36	0.78
x6	0.36	0.02	0.01	···	···	0.32	0.02	0.22
x7	0.2	0.2	0.21	···	···	0.21	0.02	0.11
x8	0.1	0.03	0.32	···	···	0.33	0.51	0.44
x9	0.32	0.1	0.2	···	···	0.06	0.66	0
x10	0.24	0	0.02	···	···	0.6	0.03	0.33
x11	0.12	0.45	0.44	···	···	0.64	0.45	1
x12	0.36	0.58	0.12	···	···	0.73	0.58	0.67
x13	0.2	0.02	0.24	···	···	0.34	0.02	0.89
x14	0.24	0.92	0.33	···	···	0.24	0.93	0.56

How about the performance for noisy subset selection?

Noisy subset selection

Subset selection: given all items $V = \{v_1, \ldots, v_n\}$, an objective function $f: 2^V \rightarrow \mathrm{R}$ and a budget B, to find a subset $X \subseteq V$ such that

$$max_{X \subseteq V} \quad f(X) \quad s.t. \quad |X| \leq B.$$

Noise

- Multiplicative: $(1-\epsilon)f(X) \leq F(X) \leq (1+\epsilon)f(X)$
- Additive: $f(X) - \epsilon \leq F(X) \leq f(X) + \epsilon$

Theoretical analysis

Greedy algorithm & POSS:

Multiplicative noise:

$$f(X) \geq \frac{1}{1+\frac{2\epsilon B}{(1-\epsilon)\gamma}}\left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^{B}\left(1-\frac{\gamma}{B}\right)^{B}\right)\cdot OPT$$

$\varepsilon \leq 1/B$ for a constant approximation ratio

Additive noise:

$$f(X) \geq \left(1-\left(1-\frac{\gamma}{B}\right)^{B}\right)\cdot OPT - \left(\frac{2B}{\gamma}-\frac{2B}{\gamma}e^{-\gamma}\right)\epsilon$$

The noiseless approximation guarantee [Das & Kempe, ICML'11; Qian, Yu and Zhou, NIPS'15]

$$f(X) \geq \left(1-\left(1-\frac{\gamma}{B}\right)^{B}\right)\cdot OPT \geq (1-e^{-\gamma})\cdot OPT$$

a constant approximation ratio

The performance degrades largely in noisy environments

The PONSS approach

Threshold selection has theoretically been shown to be tolerant to noise [Qian, Yu and Zhou, ECJ'18]

$$f(X) \geq f(Y) \longrightarrow f(X) \geq f(Y) + \theta$$

POSS

"better"
$$X \preccurlyeq Y \Leftrightarrow \begin{cases} f(X) \geq f(Y) \\ |X| \leq |Y| \end{cases}$$

Conservative

PONSS [Qian et al., NIPS'17]

Multiplicative:
$$X \preccurlyeq Y \Leftrightarrow \begin{cases} f(X) \geq \dfrac{1+\theta}{1-\theta} f(Y) \\ |X| \leq |Y| \end{cases}$$

Additive:
$$X \preccurlyeq Y \Leftrightarrow \begin{cases} f(X) \geq f(Y) + 2\theta \\ |X| \leq |Y| \end{cases}$$

Theoretical analysis

Multiplicative noise:

PONSS $f(X) \geq \frac{1-\epsilon}{1+\epsilon}\left(1-\left(1-\frac{\gamma}{B}\right)^B\right) \cdot OPT$ Significantly better

POSS & Greedy $f(X) \geq \frac{1}{1+\frac{2\epsilon B}{(1-\epsilon)\gamma}}\left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^B\left(1-\frac{\gamma}{B}\right)^B\right) \cdot OPT$

$\gamma = 1$ (submodular), ϵ is a constant

PONSS a constant approximation ratio

POSS & Greedy $\Theta(1/B)$ approximation ratio

Theoretical analysis

Multiplicative noise:

PONSS $f(X) \geq \frac{1-\epsilon}{1+\epsilon}\left(1-\left(1-\frac{\gamma}{b}\right)^B\right) \cdot \mathrm{OPT}$ Significantly better

POSS & Greedy $f(X) \geq \frac{1}{1+\frac{2\epsilon B}{(1-\epsilon)\gamma}}\left(1-\left(\frac{1-\epsilon}{1+\epsilon}\right)^B\left(1-\frac{\gamma}{B}\right)^B\right) \cdot \mathrm{OPT}$

Additive noise:

PONSS $f(X) \geq \left(1-\left(1-\frac{\gamma}{B}\right)^B\right) \cdot \mathrm{OPT} - 2\epsilon$ better

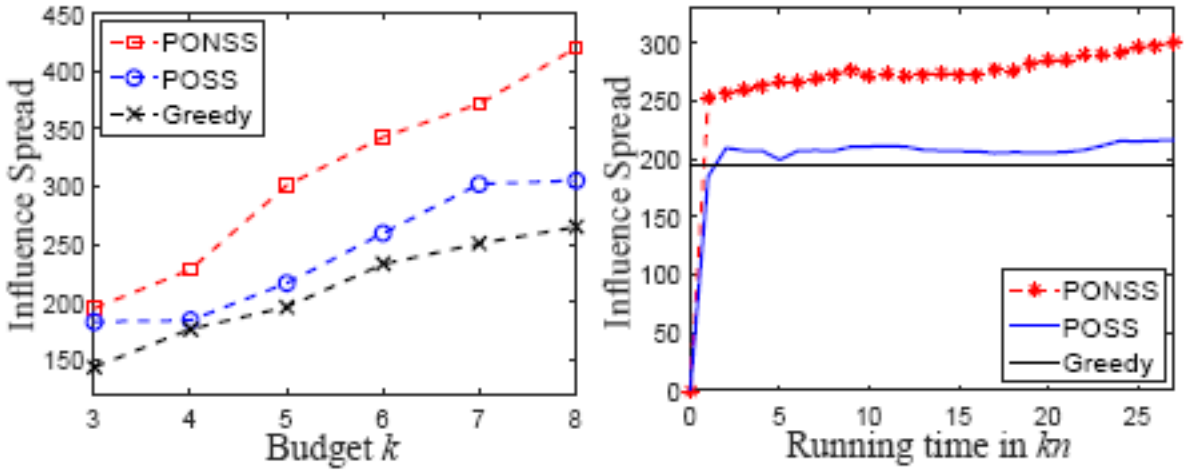
POSS & Greedy $f(X) \geq \left(1-\left(1-\frac{\gamma}{B}\right)^B\right) \cdot \mathrm{OPT} - \left(\frac{2B}{\gamma}-\frac{2B}{\gamma}e^{-\gamma}\right)\epsilon$

$\frac{2B}{\gamma}-\frac{2B}{\gamma}e^{-\gamma} \geq 2$

Experimental results - influence maximization

PONSS (red line) vs. POSS (blue line) vs. Greedy (black line):

- Noisy evaluation: the average of 10 independent Monte Carlo simulations
- The output solution: the average of 10,000 independent Monte Carlo simulations

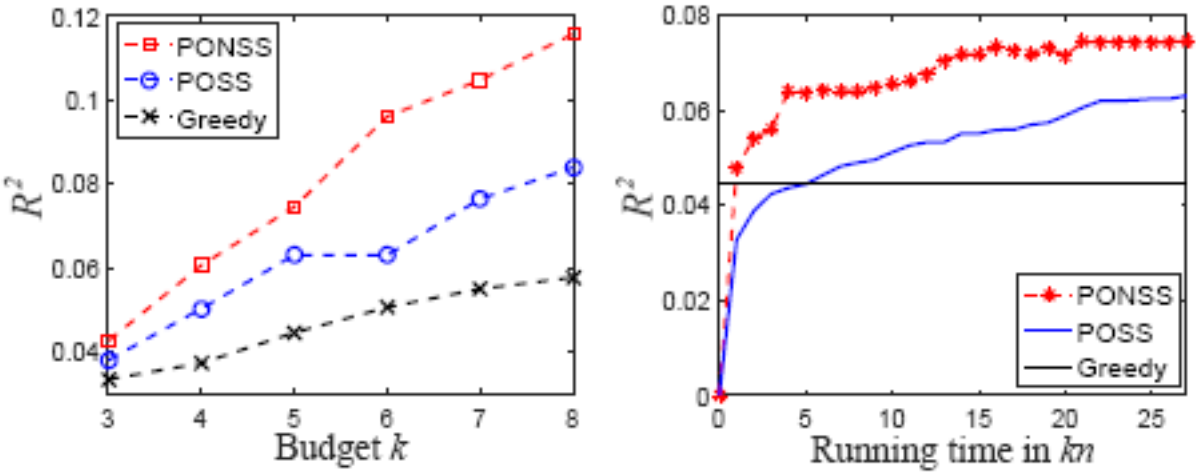


(b) *Weibo* (10,000 #nodes, 162,371 #edges)

Experimental results - sparse regression

PONSS (red line) vs. POSS (blue line) vs. Greedy (black line):

- Noisy evaluation: a random sample of 1,000 instances
- The output solution: the whole data set



(a) *protein* (24,387 #inst, 357 #feat)

Conclusion

- Running time analysis approaches for MOEAs
- Theoretical properties of MOEAs
  - Constrained optimization
  - Noisy optimization
- Multi-objective evolutionary learning algorithms
  - Selective ensemble
  - Subset selection
  - Noisy subset selection

Collaborators:

Nanjing University:

Jing-Cheng Shi

Yang Yu

Zhi-Hua Zhou

SUSTech:

Ke Tang

USTC:

Chao Bian

For details

- C. Qian, Y. Yu, Z.-H. Zhou. Pareto ensemble pruning. In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, Austin, TX, 2015.
- C. Qian, Y. Yu, Z.-H. Zhou. On constrained Boolean Pareto optimization. In: *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, Buenos Aires, Argentina, 2015.
- C. Qian, Y. Yu, Z.-H. Zhou. Subset selection by Pareto optimization. In: *Advances in Neural Information Processing Systems 28 (NIPS'15)*, Montreal, Canada, 2015.
- Y. Yu, C. Qian, Z.-H. Zhou. Switch analysis for running time analysis of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2015, 19(6): 777-792.
- C. Qian, J.-C. Shi, Y. Yu, K. Tang, Z.-H. Zhou. Subset selection under noise. In: *Advances in Neural Information Processing Systems 30 (NIPS'17)*, Long Beach, CA, 2017.
- C. Bian, C. Qian, K. Tang. A general approach to running time analysis of multi-objective evolutionary algorithms. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, Stockholm, Sweden, 2018.
- C. Qian, Y. Yu, Z.-H. Zhou. Analyzing evolutionary optimization in noisy environments. *Evolutionary Computation*, 2018, 26(1): 1-41.

Codes available at http://staff.ustc.edu.cn/~chaoqian/

THANK YOU !