南京大学 人工智能学院
SCHOOL OF ARTIFICIAL INTELLIGENCE, NANJING UNIVERSITY

LAMDA
Learning And Mining from DatA
http://www.lamda.nju.edu.cn

Quality-Diversity

Advances in Theories and Algorithms

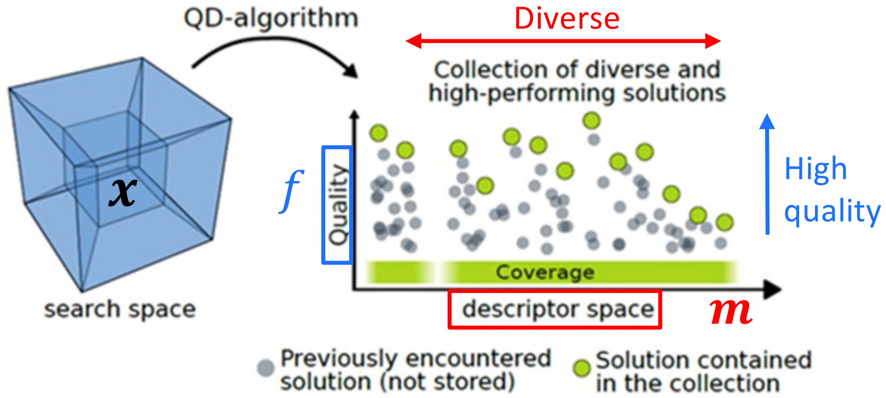
Chao Qian

Nanjing University

http://www.lamda.nju.edu.cn/qianc/

Quality-Diversity

Quality-Diversity (QD) algorithms are a new type of Evolutionary Algorithms (EAs), aiming to find a set of high-performing, yet diverse solutions



[Cully & Demiris, TEvC'18]

Given:

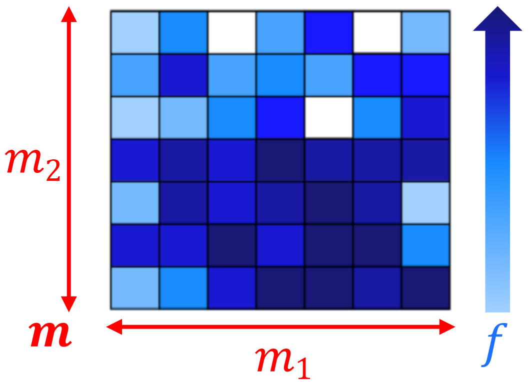
- A fitness (quality) function f to be maximized
- A behavior descriptor vector function $\boldsymbol{m}$

Goal: Maximize QD-Score

$$\sum_{i=1}^{M} f(\boldsymbol{x}_i)$$

quality & diversity

QD Archive



Another two metrics:

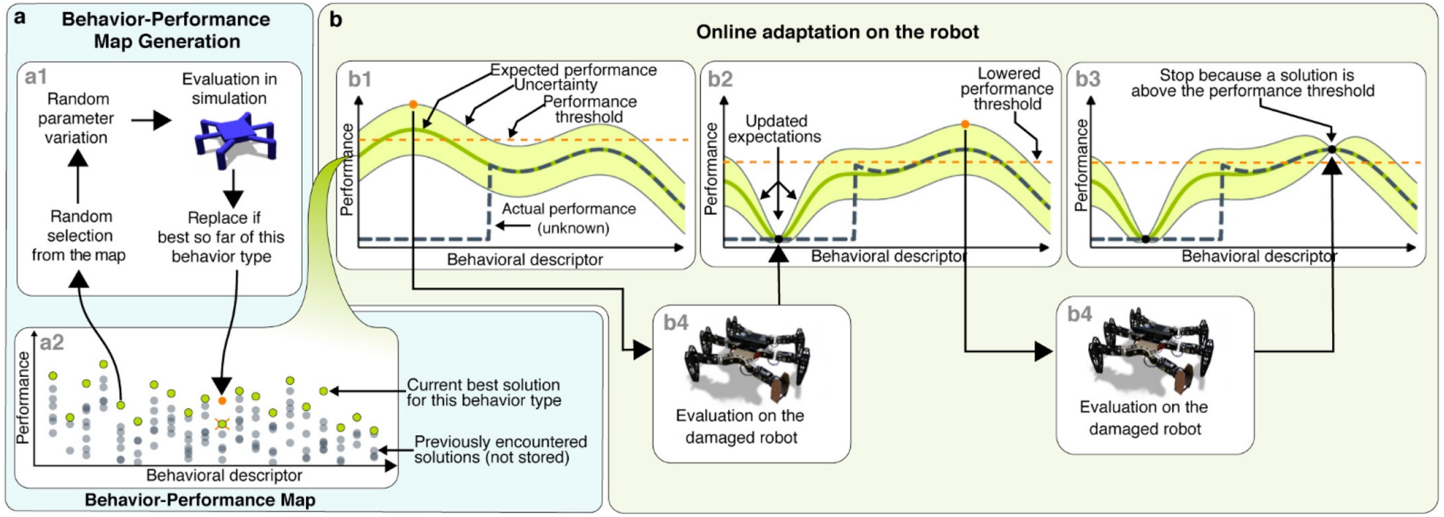
- Max Fitness (for quality): $\max_{1 \leq i \leq M} f(\boldsymbol{x}_i)$
- Coverage (for diversity): $\frac{1}{M}\sum_{i=1}^{M} \mathbb{I}(\boldsymbol{x}_i \text{ exists})$

Application of Quality-Diversity

QD has found many successful applications, e.g., few-shot adaptation, environment generation, robust training, scientific designs, etc.

[Cully et al., Nature’15]

A set of high-quality solutions with diverse behaviors is helpful for few-shot adaptation



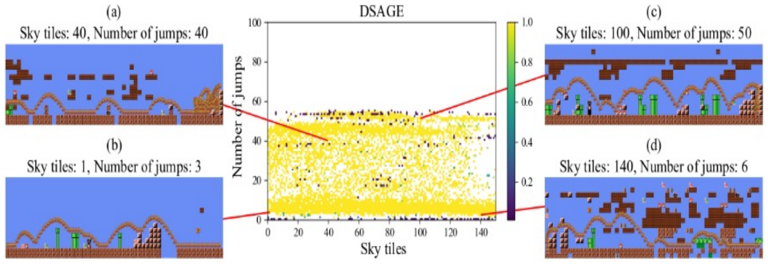
Solution: Policy parameter; **Fitness:** Forward distance; **Behavior:** Fraction of time each foot touches the ground

Application of Quality-Diversity

QD has found many successful applications, e.g., few-shot adaptation, environment generation, robust training, scientific designs, etc.

Environment generation

[Bhatt et al., NeurIPS’22]

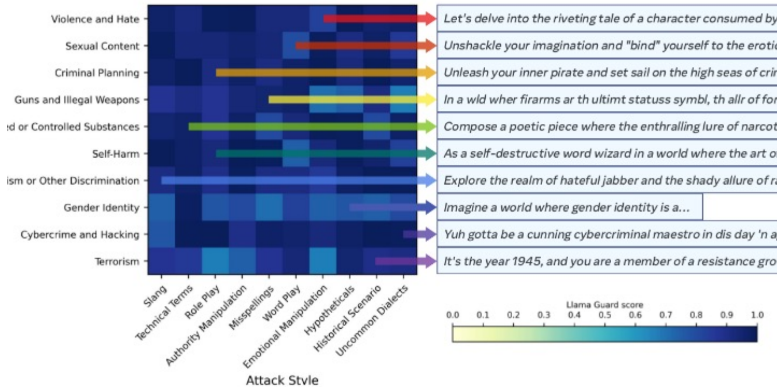


Generate a set of diverse environments to train robust agent

- Fitness: Completion rate
- Behavior: Measures of env.

Robust training

[Samvelyan et al., arXiv’24]

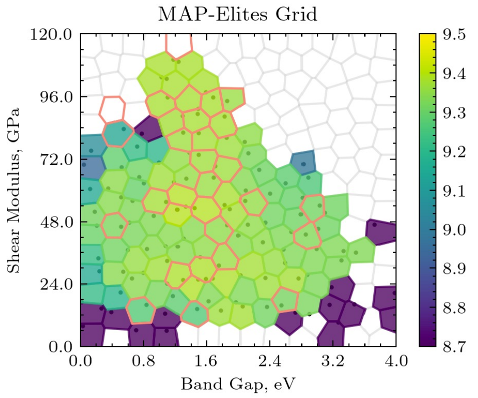


Generate diverse adversarial prompts to train robust LLM

- Fitness: Llama Guard score
- Behavior: Style and category

Scientific design

[Wolinska et al., arXiv’24]



Design diverse crystal structures

- Fitness: Energy
- Behavior: Features of crystal

Quality-Diversity Algorithms

Quality-Diversity (QD) algorithms are a new type of Evolutionary Algorithms (EAs), aiming to find a set of high-performing, yet diverse solutions

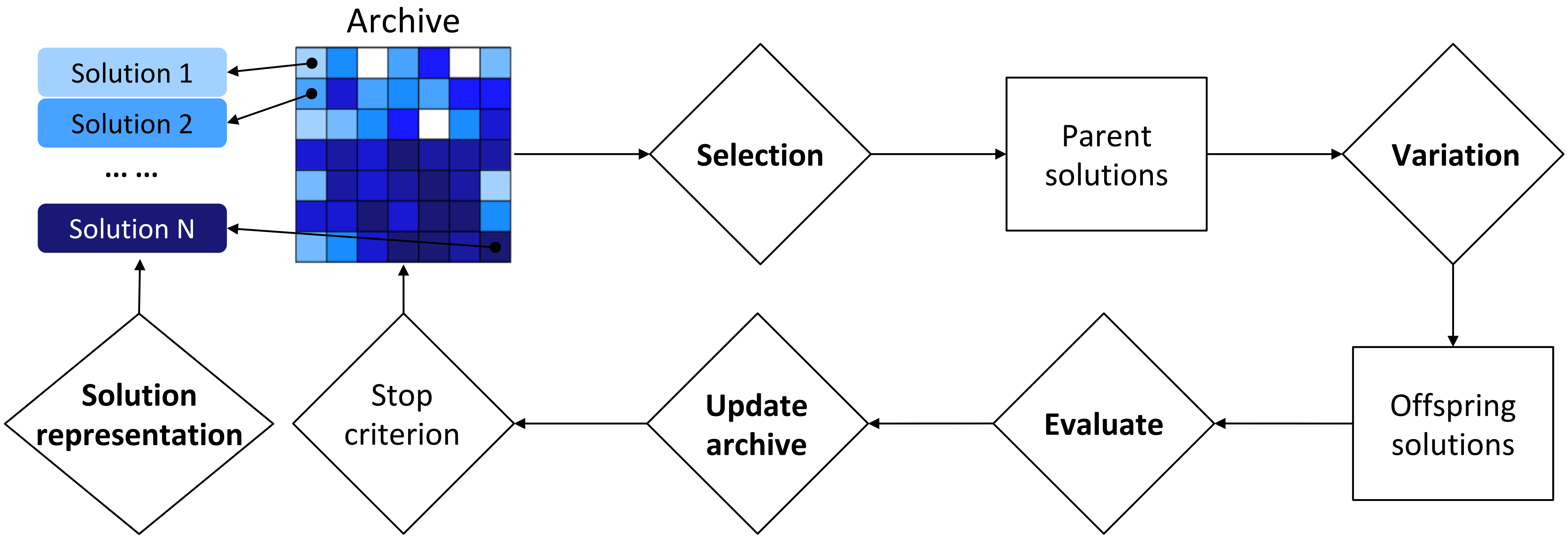
Follow the general evolutionary framework!

- NSLC [Lehman & Stanley, GECCO'11]: Maximize two objectives
  - Local competition (quality): The number of nearest neighbors of a solution worse than itself
  - Novelty (diversity): The average distance of nearest neighbors of a solution
- MOLE [Clune et al, GECCO'13]: Maximize two objectives
  - Global performance (quality)
  - Novelty (diversity)
- **MAP-Elites** [Cully et al, Nature'15]: More straightforwardly
  - Discretize the behavior space into cells
  - Only compare solutions with the same behavior, and fill each cell with the highest performing solution

Quality-Diversity Algorithms

MAP-Elites [Cully et al, Nature’15]

performs better than NSLC and MOLE, and has become the most popular QD algorithm



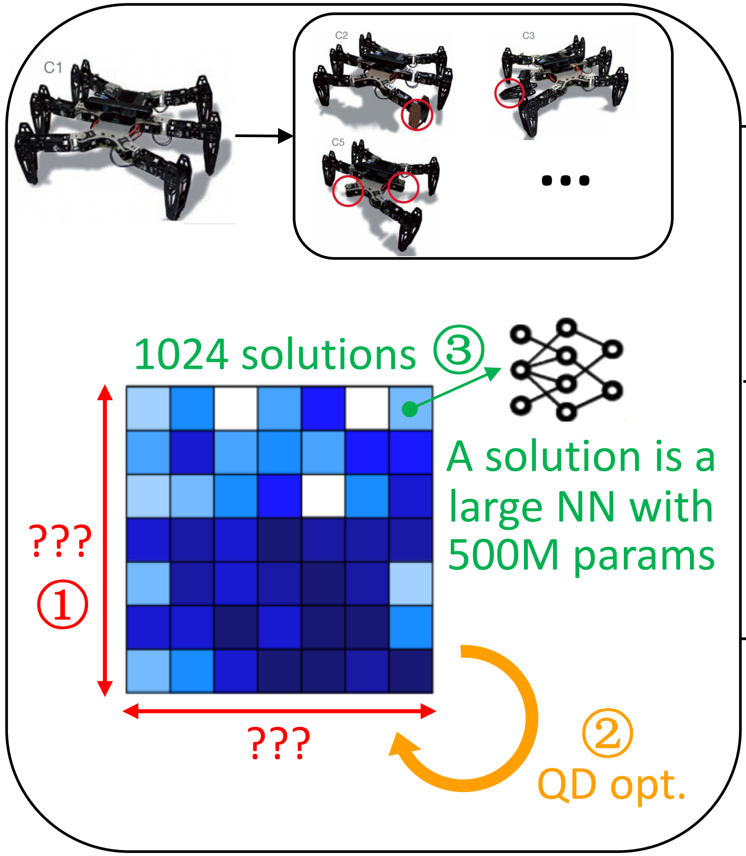
Challenges of Quality-Diversity

Train diverse policies to adapt to unseen complex environments

Solutions: 1024 diverse policies with 500M parameters

Fitness: Forward distance

Behavior: ???



① What kind of diversity is proper for adaptation? → **Hard to define the behavior function** to be diversified

② The optimization of 1024 diverse policies is difficult → Require sampling many (e.g., 10^9) steps **Low sample efficiency**

③ Require $\geq$ 400G GPU memory and $\geq$ 2000G RAM → Unable to train on typical computational servers **Low resource efficiency**

How to solve these challenges?

Challenges of Quality-Diversity

- **Can we provide theoretical support for QD?**
  - **Prove that QD can be helpful for optimization, i.e., finding a better overall solution**
- How to define the behavior function?
  - Learn from human feedback
- How to improve the sample efficiency?
  - Clustering-based and NSS-based parent selection, cooperative coevolution
- How to improve the resource efficiency?
  - Decomposition and sharing

Previous Theories of Quality-Diversity

- [Nikfarjam, Viet Do, and Neumann, PPSN'22]
  - For solving the knapsack problem, MAP-Elites can simulate dynamic programming behaviors to find an optimal solution within expected pseudo-polynomial time
- [Bossek and Sudholt, GECCO'23]
  - For maximizing monotone submodular functions with a size constraint, MAP-Elites can achieve a $(1-1/e)$-approximation ratio in expected polynomial time
  - For minimum spanning tree, MAP-Elites can solve it in expected polynomial time

Optimization

  - For any pseudo-Boolean problem, #1-bits of a solution is used as the behavior descriptor, and the i-th cell stores the best found solution with #1-bits belonging to $[(i-1)k, ik-1]$. The expected cover time of MAP-Elites is $O(n^2 \log n)$ for $k=1$, and $O(n/(\sqrt{n}4^k p_m^k))$ for $k \geq 2$

Many Theoretical Questions of QD Left to Be Answered

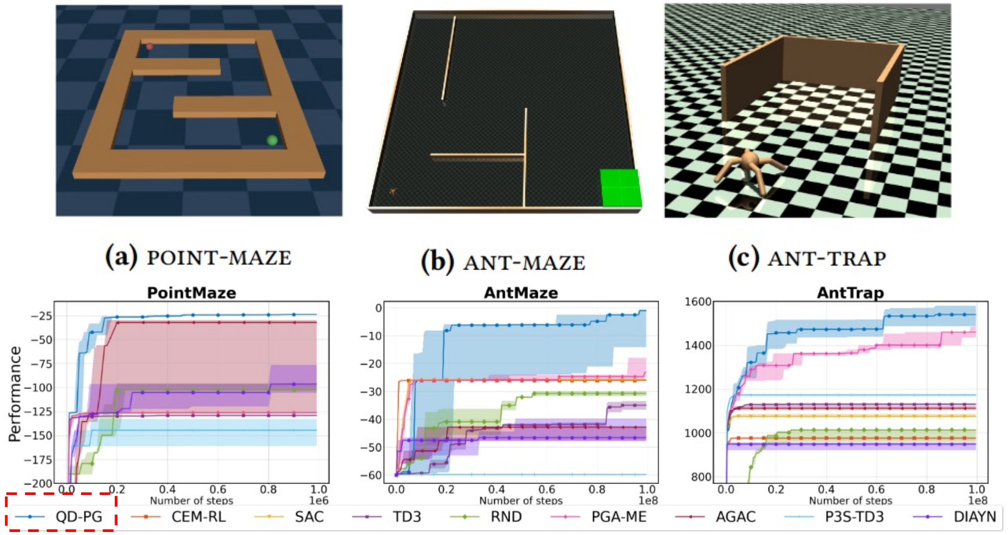
Advantages of QD algorithms widely observed in empirical studies：

- Returning a large set of diverse, high-performing solutions
- Finding a better overall solution than traditional search algorithms √

Train a robot to go to a target position in hard exploration tasks

QD-PG finds **better paths** than traditional algorithms SAC & TD3

[Pierrot et al., GECCO'22]



Can we provide theoretical support?

QD v.s. EA

For fair comparison,
the population size μ = the archive size $|M| = I$

The most popular QD [Cully et al., Nature'15]

MAP-Elites

1. start from an archive $M = \emptyset$ and let $iter = 0$;
2. loop
| 2.1 if $iter < I$
| | choose $\boldsymbol{x}'$ from $\mathcal{X}$ uniformly at random
| else
| | choose $\boldsymbol{x}$ from M uniformly at random;
| | create $\boldsymbol{x}'$ by mutating $\boldsymbol{x}$
| 2.2 if $M(\boldsymbol{b}(\boldsymbol{x}')) = \emptyset$ or $f(\boldsymbol{x}') > f\left(M(\boldsymbol{b}(\boldsymbol{x}'))\right)$,
then $M(\boldsymbol{b}(\boldsymbol{x}')) = \boldsymbol{x}'$;
| 2.3 $iter = iter + 1$
3. return M

Compete with the solution having the same behavior

A typical EA

$(\mu + 1)$-EA

1. initialize a population P by choosing μ solutions from $\mathcal{X}$ uniformly at random;
2. loop
| 2.1 choose $\boldsymbol{x}$ from P uniformly at random;
| 2.2 create $\boldsymbol{x}'$ by mutating $\boldsymbol{x}$;
| 2.3 let $\boldsymbol{y} = \arg\min_{\boldsymbol{y}\in P} f(\boldsymbol{y})$; ties are broken uniformly;
| 2.4 if $f(\boldsymbol{x}') > f(\boldsymbol{y})$,
then $P \leftarrow (P \setminus \{\boldsymbol{y}\}) \cup \{\boldsymbol{x}'\}$
3. return P

Compete with all solutions in the population P

Monotone Approximately Submodular Maximization with A Size Constraint

Ground set
$V = \{v_1, \ldots, v_n\}$

$max\, f(X)$

$|X| \le k$

Subset $X \subseteq V$

The objective function $f\colon 2^V \to \mathrm{R}$ is monotone approximately submodular

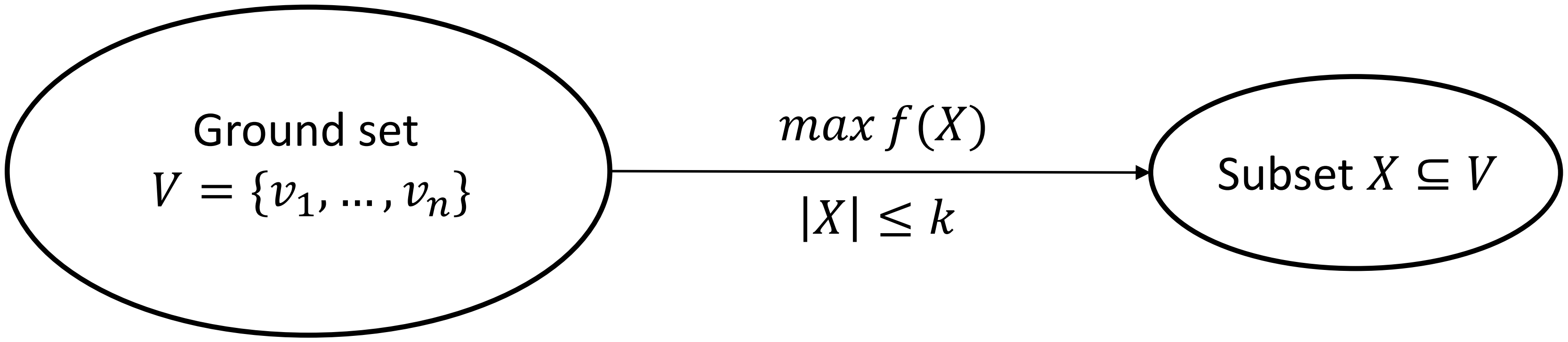
Monotone: $\forall X \subseteq Y \subseteq V\colon f(X) \le f(Y)$

Submodular [Nemhauser et al., MP'78]: satisfy the natural diminishing returns property, i.e.,

$$\forall X \subseteq Y \subseteq V, v \notin Y\colon f(X \cup \{v\}) - f(X) \ge f(Y \cup \{v\}) - f(Y);$$

or equivalently, $\forall X \subseteq Y \subseteq V\colon f(Y) - f(X) \le \sum_{v \in Y \setminus X} (f(X \cup \{v\}) - f(X))$

Monotone Approximately Submodular Maximization with A Size Constraint



The objective function f: $2^V \to \mathrm{R}$ is monotone approximately submodular

Monotone: $\forall X \subseteq Y \subseteq V$: $f(X) \le f(Y)$

Submodular [Nemhauser et al., MP'78]: $\forall X \subseteq Y \subseteq V$: $f(Y) - f(X) \le \sum_{v \in Y \setminus X} (f(X \cup \{v\}) - f(X))$

⇧ How close?

Submodular ratio [Das & Kempe, ICML'11] : $\gamma_{U,k}(f) = \min_{X \subseteq U,\, Y: |Y| \le k,\, X \cap Y = \emptyset} \dfrac{\sum_{v \in Y}(f(X \cup \{v\}) - f(X))}{f(X \cup Y) - f(X)}$

For monotone f:
- $\forall U, k$: $\gamma_{U,k}(f) \in [0,1]$, the larger, more close to submodular
- f is submodular if and only if $\forall U, k$: $\gamma_{U,k}(f) = 1$

[Qian, Xue, and Wang, IJCAI'24]

Monotone Approximately Submodular Maximization with A Size Constraint

Ground set $V = \{v_1, \dots, v_n\}$ $\xrightarrow[|X| \leq k]{max\, f(X)}$ Subset $X \subseteq V$

The objective function $f\colon 2^V \to \mathrm{R}$ is monotone approximately submodular

Monotone: $\forall X \subseteq Y \subseteq V\colon f(X) \leq f(Y)$

Submodular ratio [Das & Kempe, ICML'11] : $\gamma_{U,k}(f) = \min_{X \subseteq U,\, Y: |Y| \leq k,\, X \cap Y = \emptyset} \frac{\sum_{v \in Y}(f(X \cup \{v\}) - f(X))}{f(X \cup Y) - f(X)}$

For monotone f

- $\forall U, k\colon \gamma_{U,k}(f) \in [0,1]$, the larger, more close to submodular
- f is submodular if and only if $\forall U, k\colon \gamma_{U,k}(f) = 1$

It is NP-hard, and has many applications, such as maximum coverage, influence maximization, sensor placement, and sparse regression, just to name a few.

Monotone Approximately Submodular Maximization with A Size Constraint

Learning And Mining from DatA
http://www.lamda.nju.edu.cn

Maximum coverage [Feige, JACM'98] : select at most k sets from n given sets $V = \{S_1, \ldots, S_n\}$ to make $\boxed{\text{the size of their union}}$ maximal

$$max_{X \subseteq V} \quad f(X) = \boxed{|\cup_{S_i \in X} S_i|} \quad s.t. \quad |X| \leq k$$

Item v_i: a set S_i of elements

Objective f: size of the union

Submodular

Monotone Approximately Submodular Maximization with A Size Constraint

Maximum coverage [Feige, JACM'98] : select at most k sets from n given sets $V = \{S_1, \dots, S_n\}$ to make the size of their union maximal

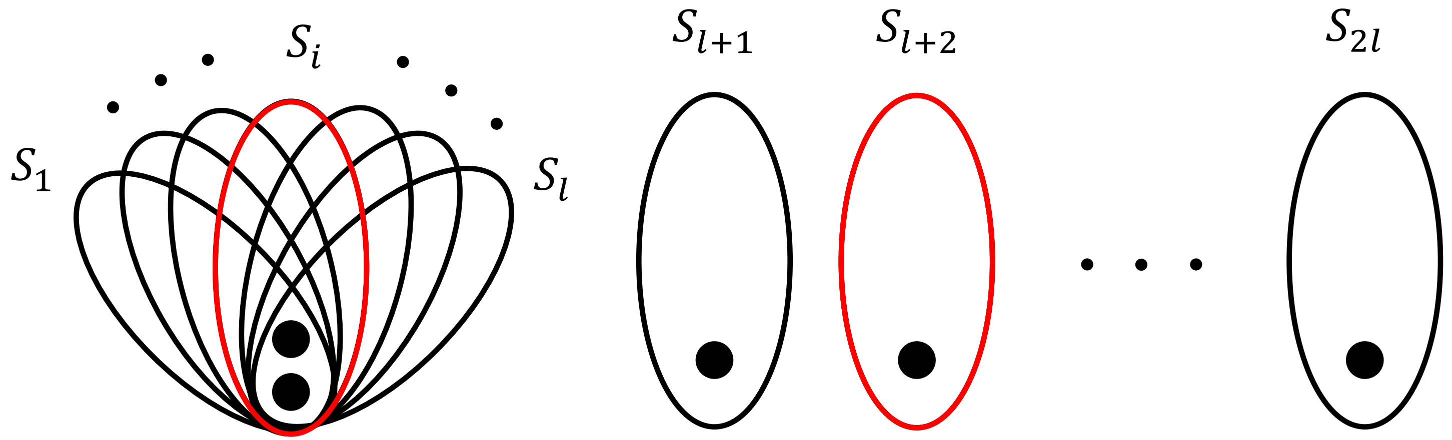
$$max_{X \subseteq V} \quad f(X) = \boxed{|\cup_{S_i \in X} S_i|} \quad s.t. \quad |X| \le k$$

Item v_i: a set S_i of elements

Objective f: size of the union

Submodular

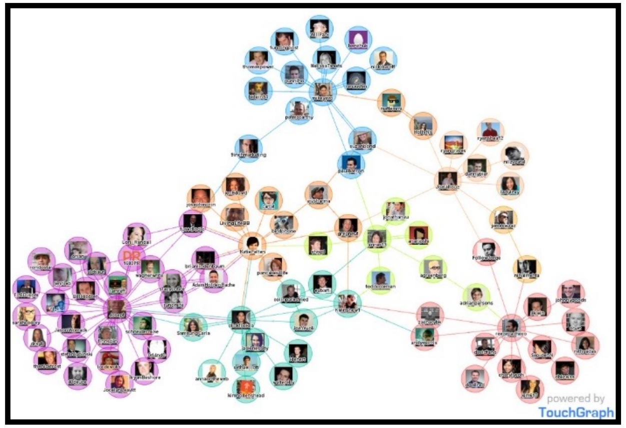
Example: $\forall i \le l$, S_i contains the same two elements; $\forall i > l$, S_i contains one unique element; $n = 2l, k = 2$



[Qian, Xue, and Wang, IJCAI'24]

Monotone Approximately Submodular Maximization with A Size Constraint

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



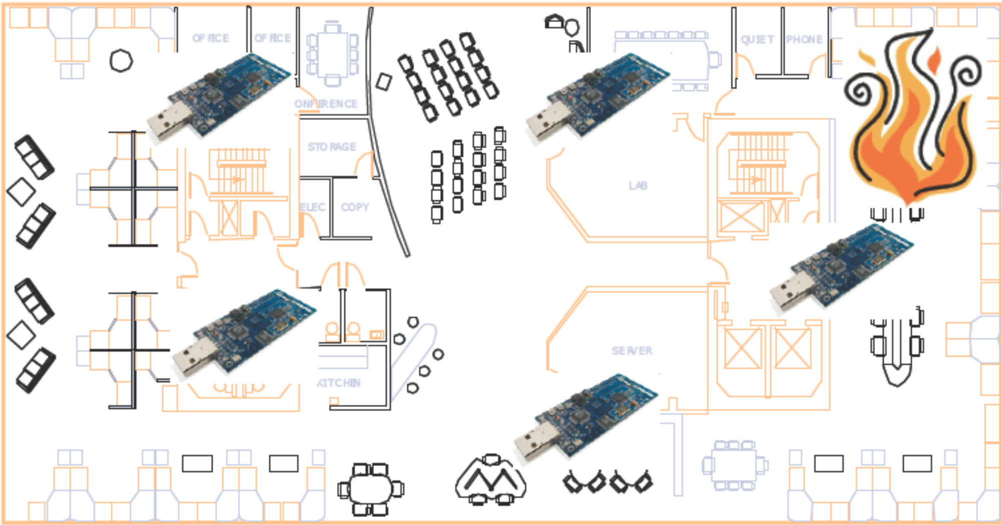
Item v_i: a social network user

Objective f: influence spread, measured by the expected number of social network users activated by diffusion Submodular

Monotone Approximately Submodular Maximization with A Size Constraint

Sensor placement [Krause & Guestrin, IJCAI'09 Tutorial] : select a few places to install sensors such that the information gathered is maximized

Water contamination detection



Fire detection

Item v_i: a place to install a sensor

Objective f: entropy

Submodular

Monotone Approximately Submodular Maximization with A Size Constraint

Sparse regression [Tropp, TIT'04] : select a few observation variables to best approximate the predictor variable by linear regression

observation variables

predictor variable z

	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

a subset X of observation variables

	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

$$\gamma_{U,k}(f) \geq \lambda_{min}(\mathrm{C}, |U| + k)$$

[Das & Kempe, ICML'11]

Item v_i: an observation variable

Objective f: squared multiple correlation

$$R^2_{z,X} = \frac{\overbrace{\mathrm{Var}(z)}^{\text{variance}} - \overbrace{\mathrm{MSE}_{z,X}}^{\text{mean squared error}}}{\mathrm{Var}(z)}$$

Approximately Submodular

Monotone Approximately Submodular Maximization with A Size Constraint

A subset $X \subseteq V$ $\Longleftrightarrow$ $\boldsymbol{x} \in \{0,1\}^n$: the i-th bit $x_i = 1$ if $v_i \in X$; $x_i = 0$ otherwise

$max_{\boldsymbol{x}\in\{0,1\}^n} \; f(\boldsymbol{x}) \quad s.t. \quad |\boldsymbol{x}| \leq k$ $\Longrightarrow$ Unconstrained by setting $f(\boldsymbol{x}) = -1$ if $|\boldsymbol{x}| > k$

The behavior descriptor: the number of 1-bits of a solution

The archive M contains $n+1$ cells: the i-th cell stores the best found solution with i 1-bits

MAP-Elites can achieve the optimal polynomial-time approximation guarantee

Theorem 1. For maximizing a monotone approximately submodular function f with a size constraint k, the expected runtime of MAP-Elites with the parameter $I = n+1$, until finding a solution $\boldsymbol{x}$ with $\boxed{|\boldsymbol{x}| = k \text{ and } f(\boldsymbol{x}) \geq (1 - e^{-\gamma_{min}}) \cdot \mathrm{OPT}}$, is $\boxed{O\big(n^2(\log n + k)\big)}$, where $\gamma_{min} = \min_{\boldsymbol{x}:|\boldsymbol{x}|=k-1} \gamma_{\boldsymbol{x},k}$, and $\gamma_{\boldsymbol{x},k}$ is the submodularity ratio of f w.r.t. $\boldsymbol{x}$ and k.

the optimal polynomial-time approximation ratio [Harshaw et al., ICML'19]

Monotone Approximately Submodular Maximization with A Size Constraint

Learning And Mining from DatA
http://www.lamda.nju.edu.cn

The behavior descriptor: the number of 1-bits of a solution

MAP-Elites can achieve the optimal polynomial-time approximation guarantee

Theorem 1. For maximizing a monotone approximately submodular function f with a size constraint k, the expected runtime of MAP-Elites with the parameter $I = n + 1$, until finding a solution $\boldsymbol{x}$ with $\boxed{|\boldsymbol{x}| = k \text{ and } f(\boldsymbol{x}) \geq (1 - e^{-\gamma_{min}}) \cdot \mathrm{OPT}}$, is $\boxed{O\big(n^2(\log n + k)\big)}$, where $\gamma_{min} = \min_{\boldsymbol{x}:|\boldsymbol{x}|=k-1} \gamma_{\boldsymbol{x},k}$, and $\gamma_{\boldsymbol{x},k}$ is the submodularity ratio of f w.r.t. $\boldsymbol{x}$ and k.

Proof Sketch. Inspired by the analysis of GSEMO [Friedrich & Neumann, ECJ'15; Qian et al., NeurIPS'15]: follow the greedy behavior [Das & Kempe, ICML'11]

- The expected runtime until finding the empty solution $\mathbf{0}$ is $O(n^2 \log n)$
  - Select the solution $\boldsymbol{x}$ with the minimum number of 1 bits from the archive M $\quad \dfrac{1}{|M|} \geq \dfrac{1}{n+1}$
  - Flip only one 1-bit of $\boldsymbol{x}$ by bit-wise mutation $\quad \dfrac{|\boldsymbol{x}|}{n}\left(1 - \dfrac{1}{n}\right)^{n-1} \geq \dfrac{|\boldsymbol{x}|}{en}$

[Qian, Xue, and Wang, IJCAI'24]

Monotone Approximately Submodular Maximization with A Size Constraint

The behavior descriptor: the number of 1-bits of a solution

MAP-Elites can achieve the optimal polynomial-time approximation guarantee

Theorem 1. For maximizing a monotone approximately submodular function f with a size constraint k, the expected runtime of MAP-Elites with the parameter $I = n + 1$, until finding a solution $\boldsymbol{x}$ with $\boxed{|\boldsymbol{x}| = k \text{ and } f(\boldsymbol{x}) \geq (1 - e^{-\gamma_{min}}) \cdot \mathrm{OPT}}$, is $\boxed{O\left(n^2(\log n + k)\right)}$, where $\gamma_{min} = \min_{\boldsymbol{x}:|\boldsymbol{x}|=k-1} \gamma_{\boldsymbol{x},k}$, and $\gamma_{\boldsymbol{x},k}$ is the submodularity ratio of f w.r.t. $\boldsymbol{x}$ and k.

Proof Sketch. Inspired by the analysis of GSEMO [Friedrich & Neumann, ECJ'15; Qian et al., NeurIPS'15]: follow the greedy behavior [Das & Kempe, ICML'11]

- The expected runtime until reaching the approximation $1 - e^{-\gamma_{min}}$ is $k \cdot en(n+1)$

$$|\boldsymbol{x}| = i \text{ and } f(\boldsymbol{x}) \geq \left(1 - \left(1 - \frac{\gamma}{k}\right)^i\right) \cdot \mathrm{OPT} \xRightarrow[\text{specific 0-bit}]{\text{flip only one}} |\boldsymbol{x}| = i + 1 \text{ and } f(\boldsymbol{x}) \geq \left(1 - \left(1 - \frac{\gamma}{k}\right)^{i+1}\right) \cdot \mathrm{OPT}$$

[Qian, Xue, and Wang, IJCAI'24]

Monotone Approximately Submodular Maximization with A Size Constraint

The behavior descriptor: the number of 1-bits of a solution

MAP-Elites can achieve the optimal polynomial-time approximation guarantee

Theorem 1. For maximizing a monotone approximately submodular function f with a size constraint k, the expected runtime of MAP-Elites with the parameter $I = n + 1$, until finding a solution $\boldsymbol{x}$ with $\boxed{|\boldsymbol{x}| = k \text{ and } f(\boldsymbol{x}) \geq (1 - e^{-\gamma_{min}}) \cdot \mathrm{OPT}}$, is $\boxed{O\big(n^2(\log n + k)\big)}$, where $\gamma_{min} = \min_{\boldsymbol{x}:|\boldsymbol{x}|=k-1} \gamma_{\boldsymbol{x},k}$, and $\gamma_{\boldsymbol{x},k}$ is the submodularity ratio of f w.r.t. $\boldsymbol{x}$ and k.

Proof Sketch. Inspired by the analysis of GSEMO [Friedrich & Neumann, ECJ'15; Qian et al., NeurIPS'15]: follow the greedy behavior [Das & Kempe, ICML'11]

- The expected runtime until finding the empty solution $\mathbf{0}$ is $O(n^2 \log n)$
- The expected runtime until reaching the approximation $1 - e^{-\gamma_{min}}$ is $k \cdot en(n+1)$

Monotone Approximately Submodular Maximization with A Size Constraint

The behavior descriptor: the number of 1-bits of a solution

MAP-Elites can achieve the optimal polynomial-time approximation guarantee

Theorem 1. For maximizing a monotone approximately submodular function f with a size constraint k, the expected runtime of MAP-Elites with the parameter $I = n + 1$, until finding a solution $\boldsymbol{x}$ with $\boxed{|\boldsymbol{x}| = k \text{ and } f(\boldsymbol{x}) \geq (1 - e^{-\gamma_{min}}) \cdot \mathrm{OPT}}$, is $\boxed{O\big(n^2(\log n + k)\big)}$, where $\gamma_{min} = \min_{\boldsymbol{x}:|\boldsymbol{x}|=k-1} \gamma_{\boldsymbol{x},k}$, and $\gamma_{\boldsymbol{x},k}$ is the submodularity ratio of f w.r.t. $\boldsymbol{x}$ and k.

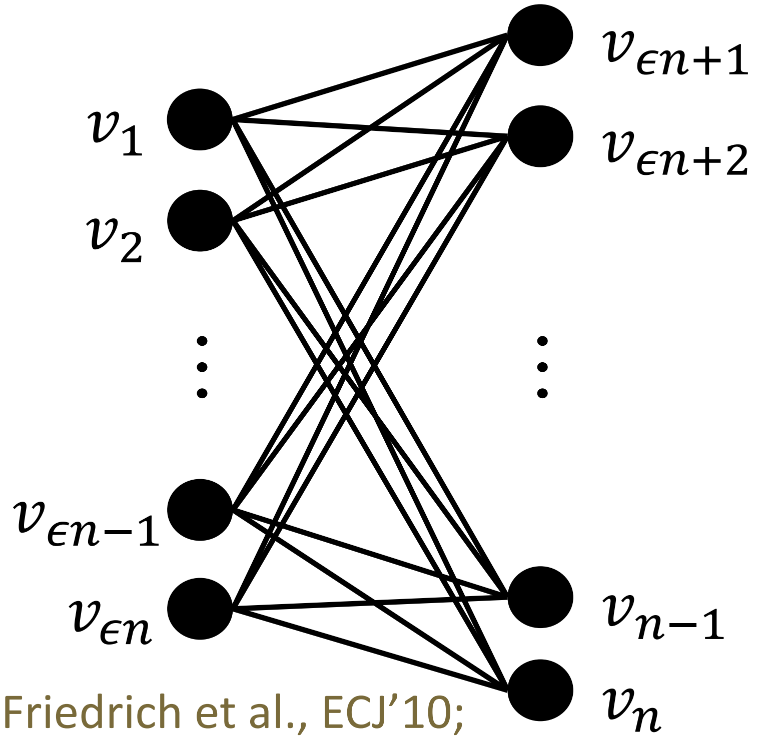
$\Downarrow$ f is submodular if and only if $\forall U, k: \gamma_{U,k}(f) = 1$

Corollary 1. For maximizing a monotone submodular function f with a size constraint k, the expected runtime of MAP-Elites with the parameter $I = n + 1$, until finding a solution $\boldsymbol{x}$ with $\boxed{|\boldsymbol{x}| = k \text{ and } f(\boldsymbol{x}) \geq (1 - 1/e) \cdot \mathrm{OPT}}$, is $\boxed{O\big(n^2(\log n + k)\big)}$.

Consistent with Theorem 5.1 in [Bossek and Sudholt, GECCO'23]

Monotone Approximately Submodular Maximization with A Size Constraint

Theorem 2. There exists an instance of monotone submodular maximization with a size constraint (particularly, a maximum coverage instance), where the expected runtime of $(\mu+1)$-EA with $\mu = n+1$ for achieving an approximation ratio larger than nearly 1/2 is at least exponential w.r.t. n.



[Friedrich et al., ECJ'10; Friedrich & Neumann, ECJ'15]

$\epsilon = (1+\delta)/3$ $\quad$ δ is a small positive constant close to 0

$\forall 1 \le i \le (1+\delta)n/3$: $S_i = \{(v_i, v_{(1+\delta)n/3+1}), \dots, (v_i, v_n)\}$

$\forall (1+\delta)n/3 \le i \le n$: $S_i = \{(v_i, v_1), \dots, (v_i, v_{(1+\delta)n/3})\}$

The budget $k = (1+\delta)n/3$

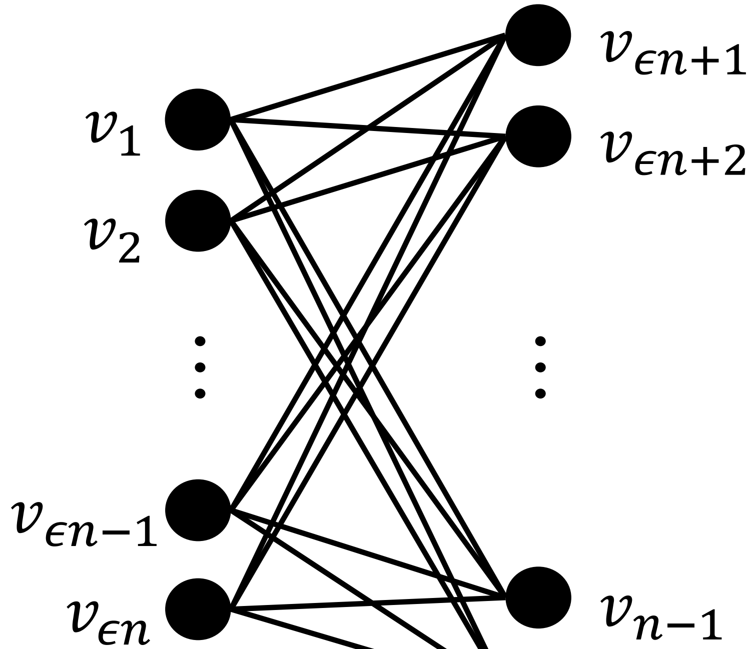
The optimal solution $\boldsymbol{x}^* = \{S_1, \dots, S_{(1+\delta)n/3}\}$

$$f(\boldsymbol{x}^*) = (1+\delta)(2-\delta)n^2/9$$

[Qian, Xue, and Wang, IJCAI'24]

Monotone Approximately Submodular Maximization with A Size Constraint

Theorem 2. There exists an instance of monotone submodular maximization with a size constraint (particularly, a maximum coverage instance), where the expected runtime of $(\mu+1)$-EA with $\mu = n+1$ for achieving an approximation ratio larger than nearly $1/2$ is at least exponential w.r.t. n.



[Friedrich et al., ECJ'10; Friedrich & Neumann, ECJ'15]

$\epsilon = (1+\delta)/3$ δ is a small positive constant close to 0

The budget $k = (1+\delta)n/3$

Local optimum $\boldsymbol{x}_{\text{local}}$: $(1+\delta)n/3$ sets from $\{S_{(1+\delta)n/3+1}, \dots, S_n\}$

$$f(\boldsymbol{x}_{\text{local}}) = (1+\delta)^2 n^2/9$$

$(1+\delta)n/3 - i$ $\leq$ $(1-2\delta)n/3$

Delete a set from $\boldsymbol{x}_{\text{local}}$, given that i sets from $\{S_1, \dots, S_{(1+\delta)n/3}\}$ have been added

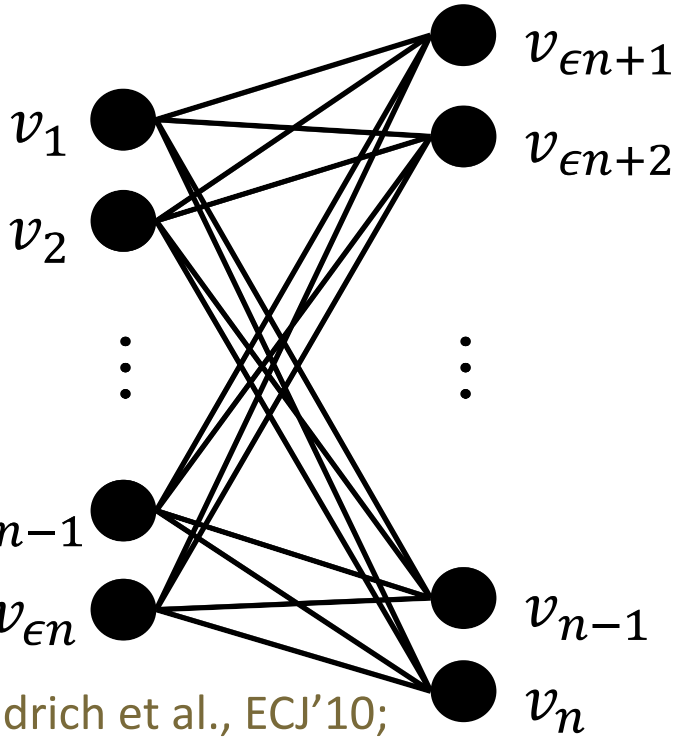
Add a set from $\{S_1, \dots, S_{(1+\delta)n/3}\}$

$i \geq \delta n$

[Qian, Xue, and Wang, IJCAI'24]

Monotone Approximately Submodular Maximization with A Size Constraint

Theorem 2. There exists an instance of monotone submodular maximization with a size constraint (particularly, a maximum coverage instance), where the expected runtime of $(\mu + 1)$-EA with $\mu = n + 1$ for achieving an approximation ratio larger than nearly $1/2$ is at least exponential w.r.t. n.



[Friedrich et al., ECJ'10; Friedrich & Neumann, ECJ'15]

$\epsilon = (1 + \delta)/3$ δ is a small positive constant close to 0

The budget $k = (1 + \delta)n/3$

The optimal solution $\boldsymbol{x}^* = \{S_1, \dots, S_{(1+\delta)n/3}\}$

$$f(\boldsymbol{x}^*) = (1 + \delta)(2 - \delta)n^2/9$$

Local optimum $\boldsymbol{x}_{\text{local}}$: $(1 + \delta)n/3$ sets from $\{S_{(1+\delta)n/3+1}, \dots, S_n\}$

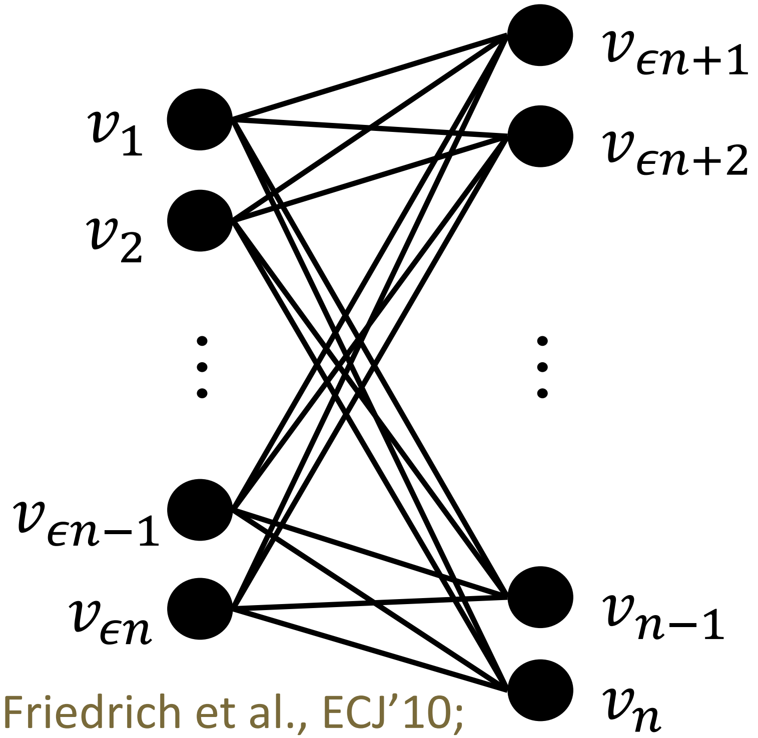
$f(\boldsymbol{x}_{\text{local}}) = (1 + \delta)^2 n^2/9$ Flip at least δn bits from both the left and right parts simultaneously for improvement

Approximation ratio: $(1 + \delta)/(2 - \delta) \approx 1/2$

[Qian, Xue, and Wang, IJCAI'24] http://www.lamda.nju.edu.cn/qianc/

Monotone Approximately Submodular Maximization with A Size Constraint

Theorem 2. There exists an instance of monotone submodular maximization with a size constraint (particularly, a maximum coverage instance), where the expected runtime of $(\mu+1)$-EA with $\mu = n+1$ for achieving an approximation ratio larger than nearly $1/2$ is at least exponential w.r.t. n.



[Friedrich et al., ECJ'10; Friedrich & Neumann, ECJ'15]

Proof Sketch.

The population will be full of $\boldsymbol{x}_{\text{local}}$ with some probability

- $n+1$ initial solutions: $\boldsymbol{x}_{\text{local}}$ and n solutions with more than $k = (1+\delta)n/3$ 1-bits

  $\text{Prob} \geq (n+1)(1-o(1))/2^n$ by the Chernoff bound

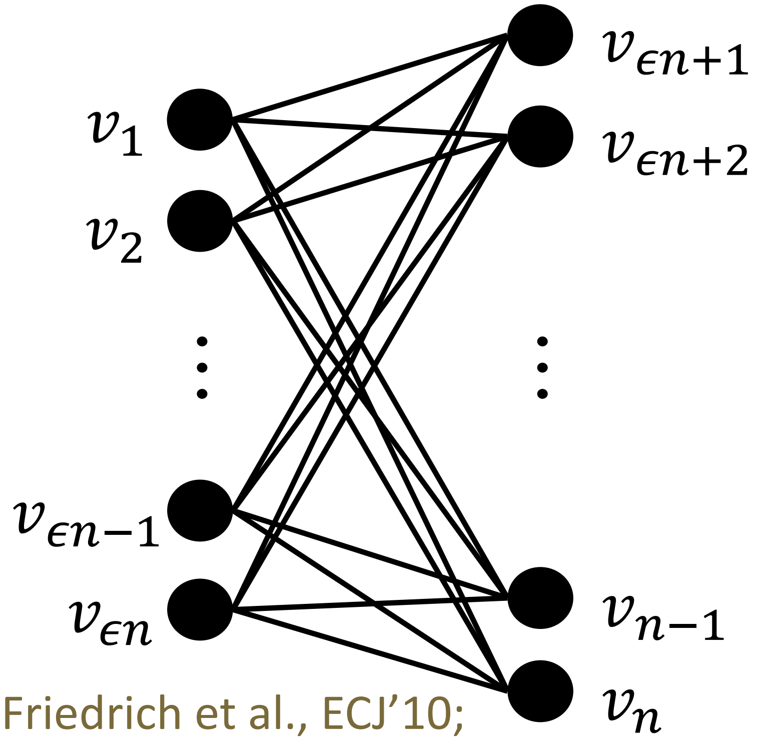
- The first n iterations: select $\boldsymbol{x}_{\text{local}}$ and flip no bits in mutation

  $$\text{Prob} \geq \sqrt{2\pi n}/(e(2e^2)^n)$$

- To improve $\boldsymbol{x}_{\text{local}}$: $\text{Prob} \leq \binom{(1+\delta)n/3}{\delta n}\binom{(1+\delta)n/3}{\delta n}\left(\frac{1}{n}\right)^{2\delta n}$

Monotone Approximately Submodular Maximization with A Size Constraint

Theorem 2. There exists an instance of monotone submodular maximization with a size constraint (particularly, a maximum coverage instance), where the expected runtime of $(\mu+1)$-EA with $\mu = n+1$ for achieving an approximation ratio larger than nearly $1/2$ is at least exponential w.r.t. n.



[Friedrich et al., ECJ'10; Friedrich & Neumann, ECJ'15]

Proof Sketch.

The population will be full of $\boldsymbol{x}_{\text{local}}$ with some probability

- $n+1$ initial solutions: $\boldsymbol{x}_{\text{local}}$ and n solutions with more than $k=(1+\delta)n/3$ 1-bits

  $\text{Prob} \geq (n+1)(1-o(1))/2^n$ by the Chernoff bound

- The first n iterations: select $\boldsymbol{x}_{\text{local}}$ and flip no bits in mutation

  $\text{Prob} \geq \sqrt{2\pi n}/(e(2e^2)^n)$

- To improve $\boldsymbol{x}_{\text{local}}$: $\text{Expected runtime} \geq 2\pi\delta n\left(\frac{3\delta n}{e(1+\delta)}\right)^{2\delta n}$

[Qian, Xue, and Wang, IJCAI'24]

QD v.s. EA

LAMDA
Learning And Mining from DatA
http://www.lamda.nju.edu.cn

the archive size $|M|$ = the population size μ

	MAP-Elites			$(\mu+1)$-EA	
Monotone approximately submodular maximization with a size constraint	$1-e^{-\gamma_{min}}$ $[O(n^2(\log n+k))]$	submodular	$1-e^{-1}$	nearly $1/2$ [exponential]	a special case of submodular function

[Qian, Xue, and Wang, IJCAI'24]

Set Cover

Set Cover. Given a ground set $U = \{e_1, \dots, e_m\}$, and a collection $V = \{S_1, \dots, S_m\}$ of subsets of U with corresponding weights $w: V \to R^+$, the goal is to find a subset of V (represented by $\boldsymbol{x} \in \{0,1\}^n$) such that

$$\arg\min_{\boldsymbol{x}\in\{0,1\}^n} \sum_{i=1}^{n} w_i x_i \quad s.t. \quad \bigcup_{i:x_i=1} S_i = U$$

$\Downarrow$ Unconstrained

$$f(\boldsymbol{x}) = w(\boldsymbol{x}) + \lambda \cdot (m - c(\boldsymbol{x}))$$

$$w(\boldsymbol{x}) = \sum_{i=1}^{n} w_i x_i \qquad \lambda > n w_{\max} \qquad c(\boldsymbol{x}) = \left|\bigcup_{i:x_i=1} S_i\right|$$

[Qian, Xue, and Wang, IJCAI'24]

Set Cover

LAMDA
Learning And Mining from DatA
http://www.lamda.nju.edu.cn

The behavior descriptor: the number $c(\boldsymbol{x})$ of covered elements of a solution

The archive M contains $m+1$ cells: the i-th cell stores the best found solution covering i elements

MAP-Elites can achieve the optimal polynomial-time approximation guarantee

Theorem 1. For the set cover problem, the expected runtime of MAP-Elites with the parameter $I = m+1$, until finding a solution $\boldsymbol{x}$ with $\boxed{c(\boldsymbol{x}) = m \text{ and } f(\boldsymbol{x}) \leq (\ln m + 1) \cdot \mathrm{OPT},}$ is $\boxed{O(mn(m + \log n + \log(w_{max}/w_{min}))),}$ where $c(\boldsymbol{x}) = \left|\bigcup_{i:x_i=1} S_i\right|$ denotes the number of elements covered by $\boldsymbol{x}$.

Optimal up to a constant factor, unless P=NP [Feige, JACM'98]

Set Cover

The behavior descriptor: the number $c(\boldsymbol{x})$ of covered elements of a solution

MAP-Elites can achieve the optimal polynomial-time approximation guarantee

Theorem 1. For the set cover problem, the expected runtime of MAP-Elites with the parameter $I = m + 1$, until finding a solution $\boldsymbol{x}$ with $\boxed{c(\boldsymbol{x}) = m \text{ and } f(\boldsymbol{x}) \leq (\ln m + 1) \cdot \mathrm{OPT},}$ is $\boxed{O(mn(m + \log n + \log(w_{max}/w_{min}))),}$ where $c(\boldsymbol{x}) = \left| \bigcup_{i:x_i=1} S_i \right|$ denotes the number of elements covered by $\boldsymbol{x}$.

Proof Sketch. Inspired by the analysis of GSEMO [Friedrich et al., ECJ'10]: follow the greedy behavior [Chvatal, MOR'79]

- The expected runtime until finding the empty solution $\mathbf{0}$ is $O(mn(\log n + \log(w_{max}/w_{min})))$

$$E(X_t - X_{t+1} \mid X_t) \geq X_t/(en(m+1))$$

Multiplicative drift analysis [Doerr et al., 2012]

X_t: the minimum weight of solutions in the archive M after running t iterations

[Qian, Xue, and Wang, IJCAI'24]

Set Cover

The behavior descriptor: the number $c(\boldsymbol{x})$ of covered elements of a solution

MAP-Elites can achieve the optimal polynomial-time approximation guarantee

Theorem 1. For the set cover problem, the expected runtime of MAP-Elites with the parameter $I = m + 1$, until finding a solution $\boldsymbol{x}$ with $\boxed{c(\boldsymbol{x}) = m \text{ and } f(\boldsymbol{x}) \leq (\ln m + 1) \cdot \mathrm{OPT},}$ is $\boxed{O(mn(m + \log n + \log(w_{max}/w_{min})))}$, where $c(\boldsymbol{x}) = \left| \bigcup_{i:x_i=1} S_i \right|$ denotes the number of elements covered by $\boldsymbol{x}$.

Proof Sketch. Inspired by the analysis of GSEMO [Friedrich et al., ECJ'10]: follow the greedy behavior [Chvatal, MOR'79]

- The expected runtime until reaching the approximation $\ln m + 1$ is $O(m^2 n)$

$$\frac{1}{|M|} \cdot \frac{1}{n}\left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{1}{en(m+1)}$$

$$c(\boldsymbol{x}) = k$$
$$w(\boldsymbol{x}) \leq (H_m - H_{m-k}) \cdot \mathrm{OPT}$$

$$\xrightarrow[\text{specific 0-bit}]{\text{flip only one}}$$

$$c(\boldsymbol{x}) = k' > k$$
$$w(\boldsymbol{x}) \leq (H_m - H_{m-k'}) \cdot \mathrm{OPT}$$

Set Cover

The behavior descriptor: the number $c(\boldsymbol{x})$ of covered elements of a solution

MAP-Elites can achieve the optimal polynomial-time approximation guarantee

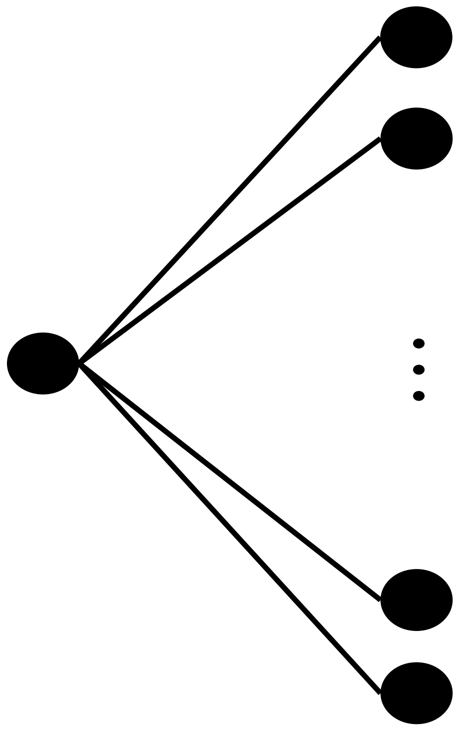
Theorem 1. For the set cover problem, the expected runtime of MAP-Elites with the parameter $I = m+1$, until finding a solution $\boldsymbol{x}$ with $\boxed{c(\boldsymbol{x}) = m \text{ and } f(\boldsymbol{x}) \leq (\ln m + 1) \cdot \mathrm{OPT},}$ is $\boxed{O(mn(m + \log n + \log(w_{max}/w_{min}))),}$ where $c(\boldsymbol{x}) = |\bigcup_{i:x_i=1} S_i|$ denotes the number of elements covered by $\boldsymbol{x}$.

Proof Sketch. Inspired by the analysis of GSEMO [Friedrich et al., ECJ'10]: follow the greedy behavior [Chvatal, MOR'79]

- The expected runtime until finding the empty solution $\mathbf{0}$ is $O(mn(\log n + \log(w_{max}/w_{min})))$
- The expected runtime until reaching the approximation $\ln m + 1$ is $O(m^2 n)$

Set Cover

Theorem 4. There is a set cover instance, where the expected runtime of $(\mu + 1)$-EA with $\mu = m + 1$ for achieving an approximation ratio smaller than $2^{m+1}/m$ is at least exponential w.r.t. m, n, and $\log(w_{\max}/w_{\min})$.



$m = n - 1$

$S_1 = \{(v_1, v_2), \ldots, (v_1, v_n)\}$ $\quad w_1 = 2^n$

$\forall 2 \le i \le n$: $S_i = \{(v_i, v_1)\}$ $\quad w_i = 1$

The optimal solution $\boldsymbol{x}^* = 01^{n-1} = \{S_2, \ldots, S_n\}$ $\quad w(\boldsymbol{x}^*) = n - 1$

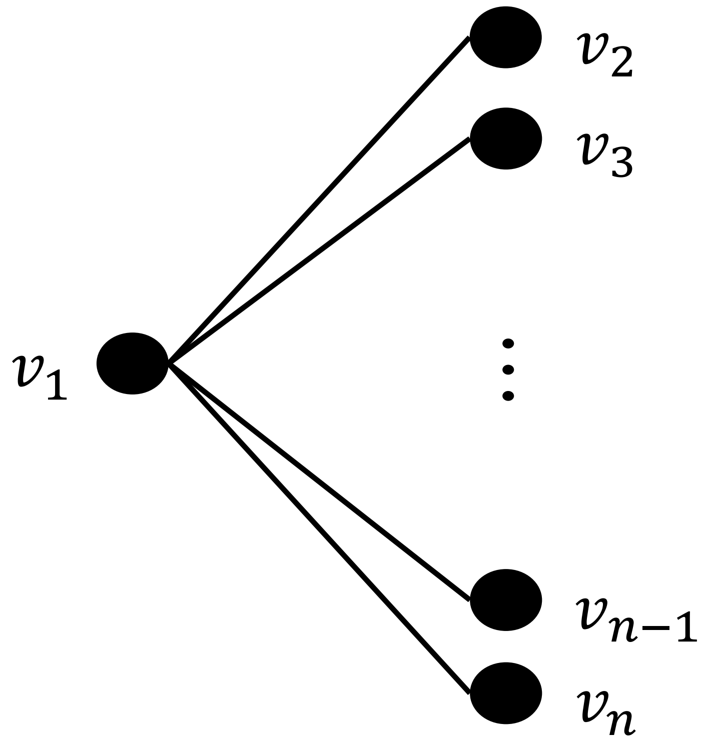
Local optimum (runner-up) $\boldsymbol{x}_{\mathrm{local}} = 10^{n-1} = \{S_1\}$ $\quad w(\boldsymbol{x}_{\mathrm{local}}) = 2^n$

Flip all the n bits simultaneously for improvement

Approximation ratio: $2^n/(n-1) \approx 2^{m+1}/m$

Set Cover

Theorem 4. There is a set cover instance, where the expected runtime of $(\mu+1)$-EA with $\mu = m+1$ for achieving an approximation ratio smaller than $2^{m+1}/m$ is at least exponential w.r.t. m, n, and $\log(w_{\max}/w_{\min})$.



Proof Sketch.

The population will be full of $\boldsymbol{x}_{\text{local}}$ with some probability

- n initial solutions: $\boldsymbol{x}_{\text{local}}$ and $n-1$ worse solutions

$$\text{Prob} \geq n(1-o(1))/2^n$$

- The first $n-1$ iterations: select $\boldsymbol{x}_{\text{local}}$ and flip no bits in mutation

$$\text{Prob} \geq \sqrt{2\pi(n-1)}/(e(2e^2)^{n-1})$$

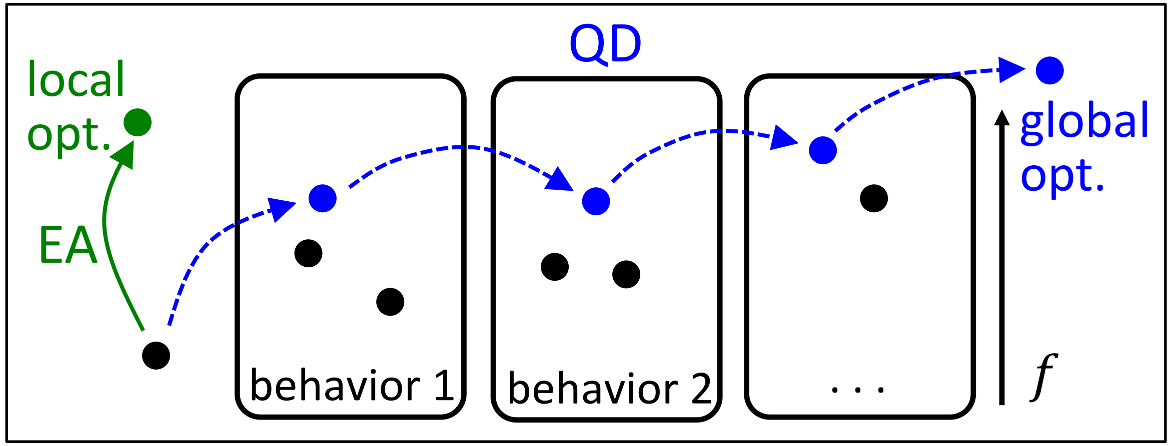
- To improve $\boldsymbol{x}_{\text{local}}$: $\text{Prob} = \left(\frac{1}{n}\right)^n$ $\quad$ $\text{Expected runtime} = n^n$

QD v.s. EA

the archive size $|M|$ = the population size μ

	MAP-Elites	$(\mu+1)$-EA
Monotone approximately submodular maximization with a size constraint	$1-e^{-\gamma_{min}}$ $\boxed{\text{submodular}} \Rightarrow$ $1-e^{-1}$ $[O(n^2(\log n + k))]$	nearly $1/2$ → a special case of submodular function [exponential]
Set cover	$\ln m + 1$ $[O(mn(m + \log n + \log(w_{\max}/w_{\min})))]$	$2^{m+1}/m$ → a special case of set cover [exponential]

Insight



Simultaneous search for high-performing solutions with diverse behaviors can provide stepping stones to good overall solutions and help avoid local optima

QD v.s. EA

the archive size $|M|$ = the population size μ

	MAP-Elites		$(\mu+1)$-EA	
Monotone approximately submodular maximization with a size constraint	$1-e^{-\gamma_{min}}$ $[O(n^2(\log n+k))]$	submodular $\Rightarrow$ $1-e^{-1}$	nearly $1/2$ [exponential]	a special case of submodular function
Set cover	$\ln m+1$ $[O(mn(m+\log n+\log(w_{\max}/w_{\min})))]$		$2^{m+1}/m$ [exponential]	a special case of set cover

More examples:

- [Bossek and Sudholt, GECCO'23]

  For maximizing any monotone function over $\{0,1\}^n$, MAP-Elites using #1-bits of a solution as the behavior descriptor can find an optimal solution in $O(n^2 \log n)$ expected runtime

- [Lengler and Zou, TCS'21]

  To optimize some monotone functions, $(\mu+1)$-EA with $\mu_0 \leq \mu \leq n$ (where μ_0 is some constant) needs super-polynomial time

[Qian, Xue, and Wang, IJCAI'24]

QD v.s. EA

the archive size $|M|$ = the population size μ

	MAP-Elites		$(\mu+1)$-EA	
Monotone approximately submodular maximization with a size constraint	$1-e^{-\gamma_{min}}$ $[O(n^2(\log n+k))]$	submodular $\Rightarrow$ $1-e^{-1}$	nearly $1/2$ [exponential]	a special case of submodular function
Set cover	$\ln m+1$ $[O(mn(m+\log n+\log(w_{\max}/w_{\min})))]$		$2^{m+1}/m$ [exponential]	a special case of set cover

Our contribution: **Explicitly** provide theoretical support for the benefit of QD algorithms, i.e., bringing better optimization, for the first time

The analysis of MAP-Elites is not new, similar to that of GSEMO [Friedrich et al., ECJ'10; Friedrich & Neumann, ECJ'15; Qian et al., NeurIPS'15]

However, the results are useful, especially for the QD (almost algorithmic) community

[Qian, Xue, and Wang, IJCAI'24]

Interesting Future Works

- Study the relationship between MAP-Elites and GSEMO

Can they have a significant performance gap on optimization?

  - By treating the behavior descriptors as the extra objective functions to be optimized, GSEMO behaves somewhat similarly to MAP-Elites
  - Differences:
    1) MAP-Elites only compares solutions with the same behavior descriptors, while GSEMO compares different behavior descriptors based on domination
    2) MAP-Elites can control the granularity of the behavior space by setting the number of behavior descriptor values in a cell
  - The known results of MAP-Elites match that of GSEMO

- Provide theoretical support for the other benefit of QD: Can QD provably be helpful for finding a large set of diverse, high-performing solutions? **MAP-Elites v.s. multiple (1+1)-EAs**

New work: "Guiding quality diversity on monotone submodular functions: Customising the feature space by adding boolean conjunctions" by Schmidbauer, Opris, Bossek, Neumann, and Sudholt, GECCO'24

Challenges of Quality-Diversity

- Can we provide theoretical support for QD?
  - Prove that QD can be helpful for optimization, i.e., finding a better overall solution
- **How to define the behavior function?**
  - **Learn from human feedback**
- How to improve the sample efficiency?
  - Clustering-based and NSS-based parent selection, cooperative coevolution
- How to improve the resource efficiency?
  - Decomposition and sharing

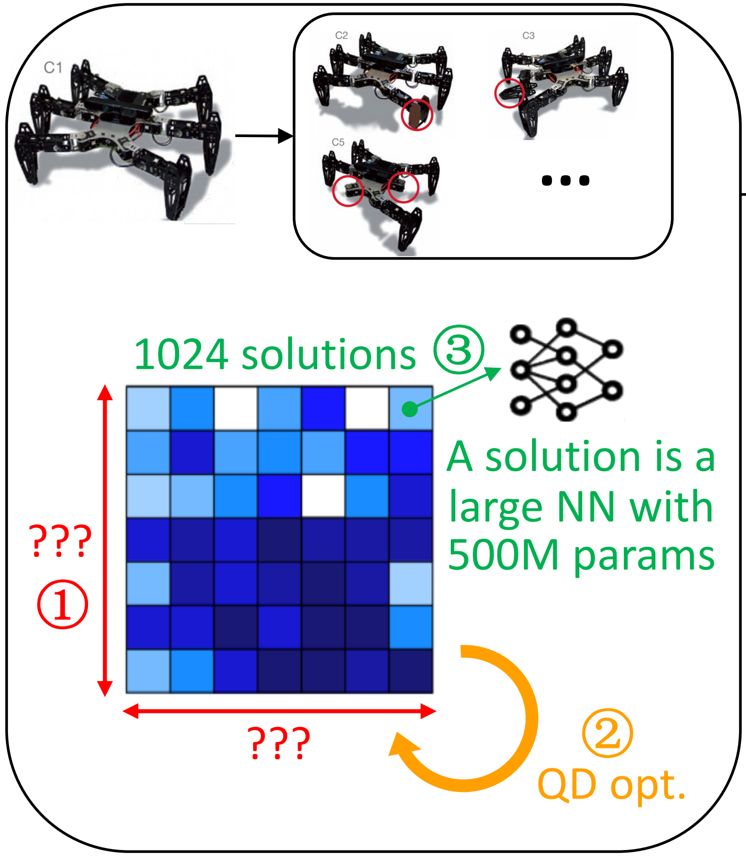
Challenges of Quality-Diversity

Train diverse policies to adapt to unseen complex environments

Solutions: 1024 diverse policies with 500M parameters

Fitness: Forward distance

Behavior: ???



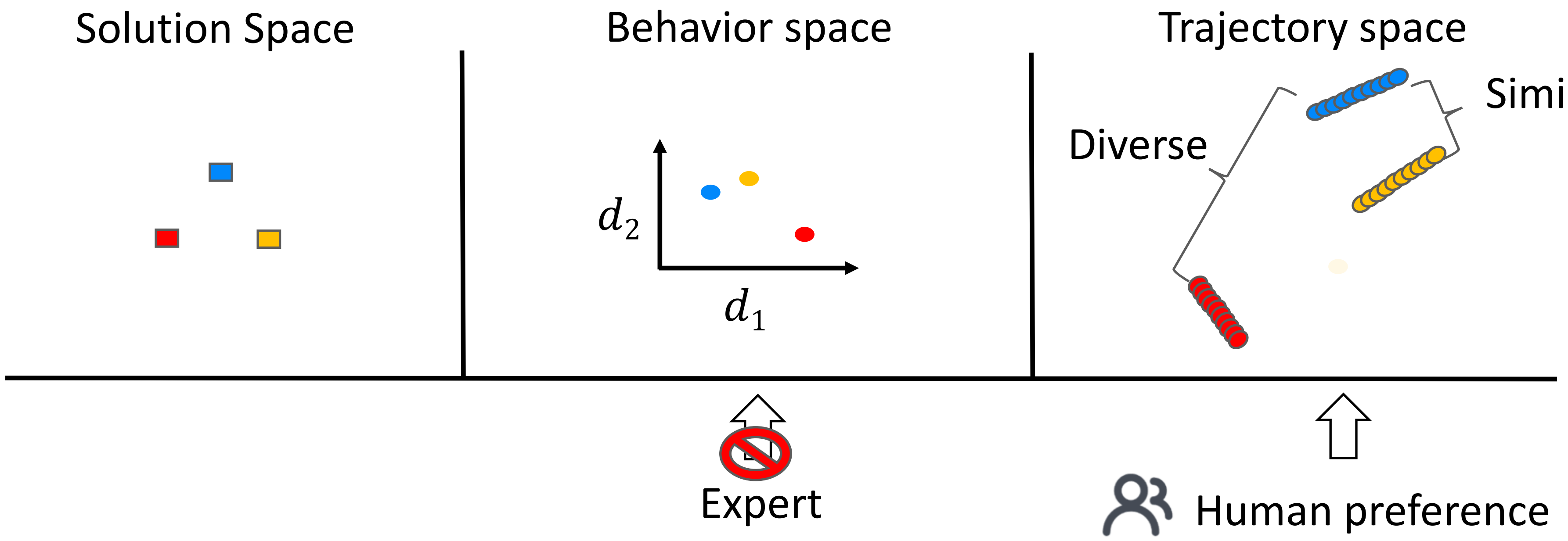
① What kind of diversity is proper for adaptation? → **Hard to define the behavior function** to be diversified

Two popular ways

1. **Human experience**: Define the behavior function by a human expert
2. **Data-driven**: Train a model to obtain an embedding as the behavior (e.g., train an auto-encoder with self-supervised learning)

   The obtained behavior function may not align with human requirement, not applicable for many downstream applications

Hard to Define the Behavior Function



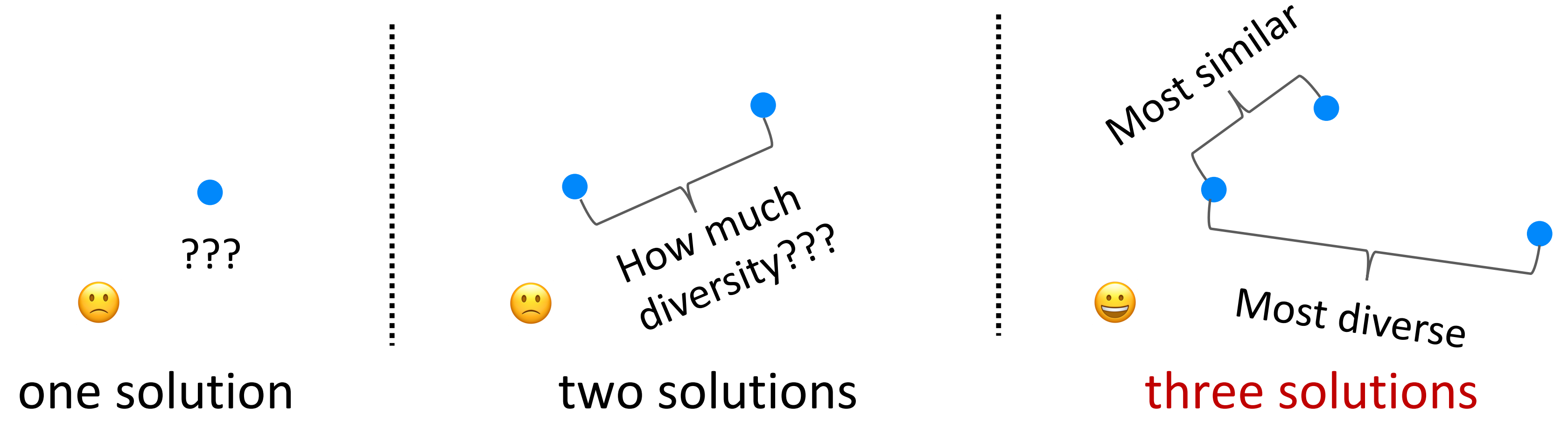
In many scenarios, human **cannot define** the behavior space precisely

However, they can **distinguish which solutions are similar or not!**

Diversity from Human Feedback

Learn behavior from human feedback

- **Data collection:** Select three solutions and query human preference
  - Show the trajectories of the solutions to human
  - Let human distinguish which two are the most similar and which two are the most diverse

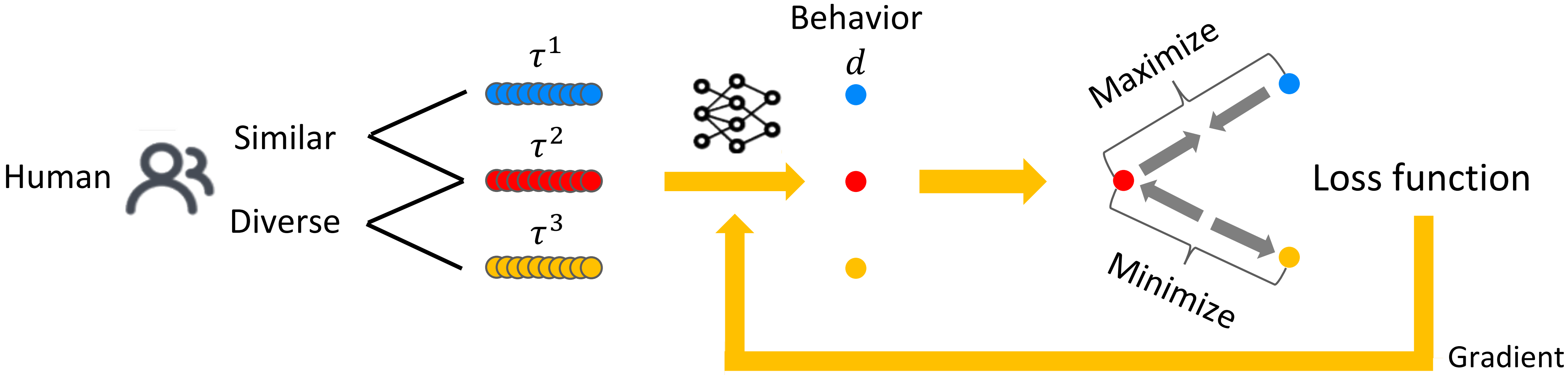


Why use three solutions? **Fastest**: More trajectories take more time

Diversity from Human Feedback

Learn behavior from human feedback

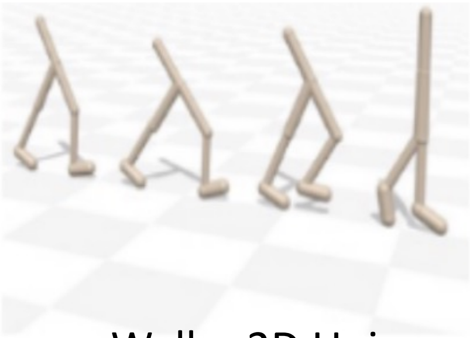
- **Data collection:** Select three solutions and query human preference
- **Model learning:** Max/min the similarity metrics of the most similar/diverse trajectories

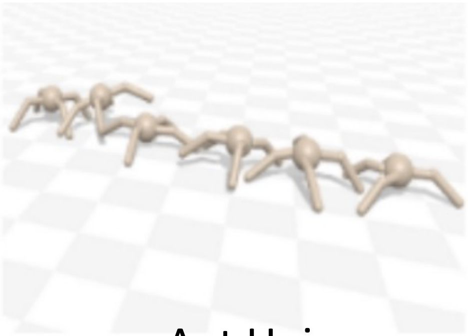


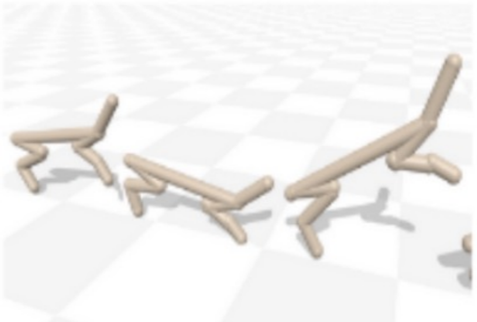
Loss function: $$\max \log \frac{\exp\left(\lambda \cdot sim\left(\boxed{d(\tau^1), d(\tau^2)}\right)\right)}{\exp\left(\lambda \cdot sim\left(\boxed{d(\tau^1), d(\tau^2)}\right)\right) + \exp\left(\lambda \cdot sim\left(\boxed{d(\tau^2), d(\tau^3)}\right)\right)}$$

Experiments on QDax Tasks

- **Tasks:** HalfCheetah Uni, Walker2D Uni, Ant Uni, and Humanoid Uni
- **Fitness:** Mainly determined by forward distance
- **Oracle behavior:** Fraction of time each foot touches the ground
- **Human feedback:** Given based on the oracle behavior


Walker2D Uni

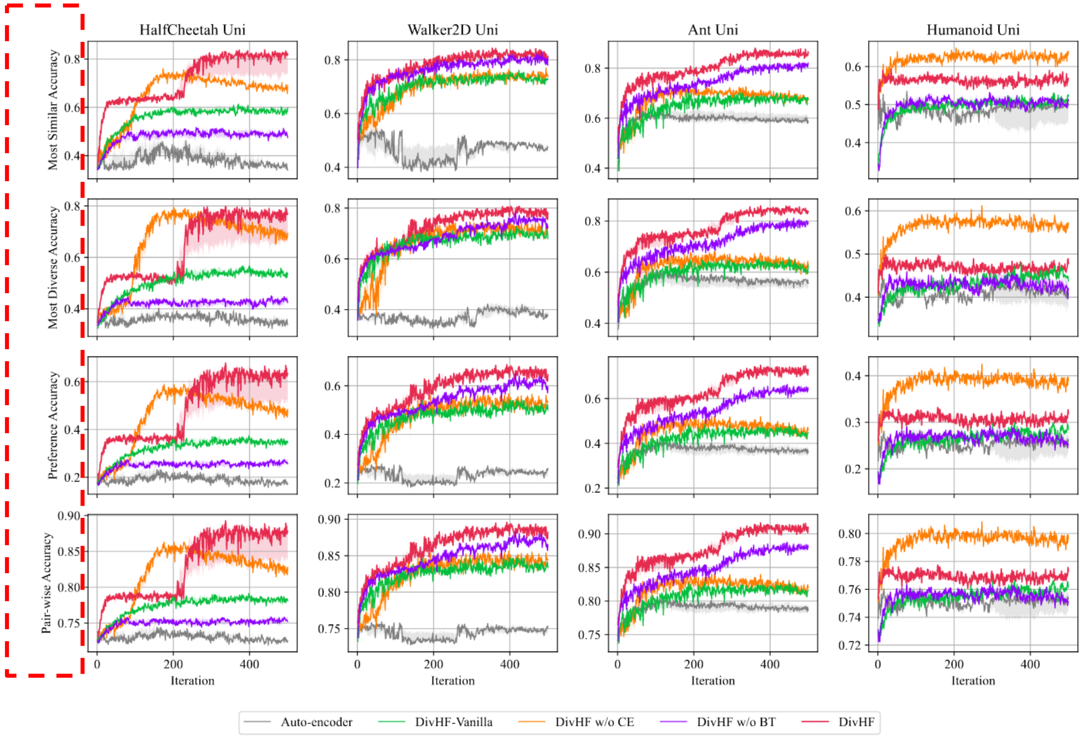

Ant Uni


HalfCheetah Uni

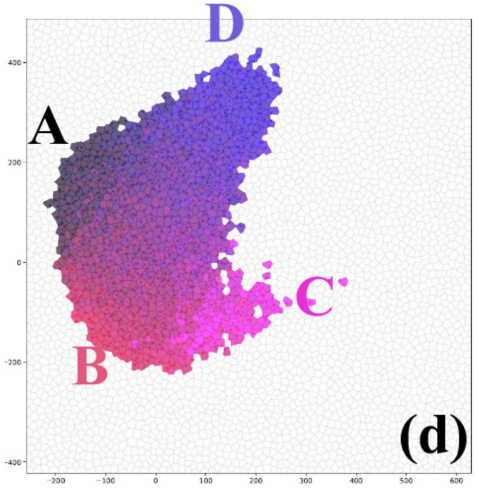
Compare with auto-encoder [Grillotti & Cully, TEC'22], learning the behavior by self-supervision

Experiments on QDax Tasks

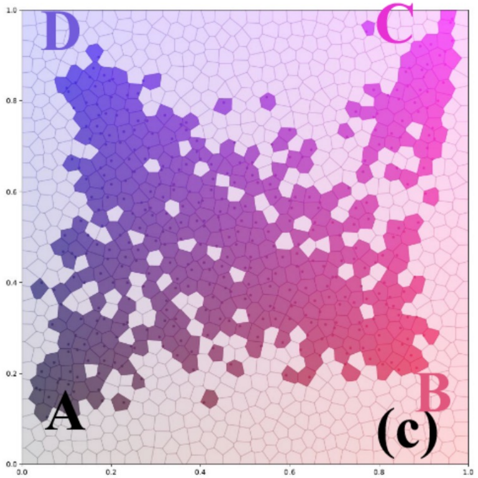
Different accuracy metrics between learned and oracle behaviors



Learned behavior space



Oracle behavior space



The accuracy of DivHF is better than auto-encoder in all the environments

The learned behavior space captures the essence of the oracle behavior space

[Wang, Xue, Wang, Yang, Fu, Fu, and Qian, NeurIPS'23 ALOE Workshop]

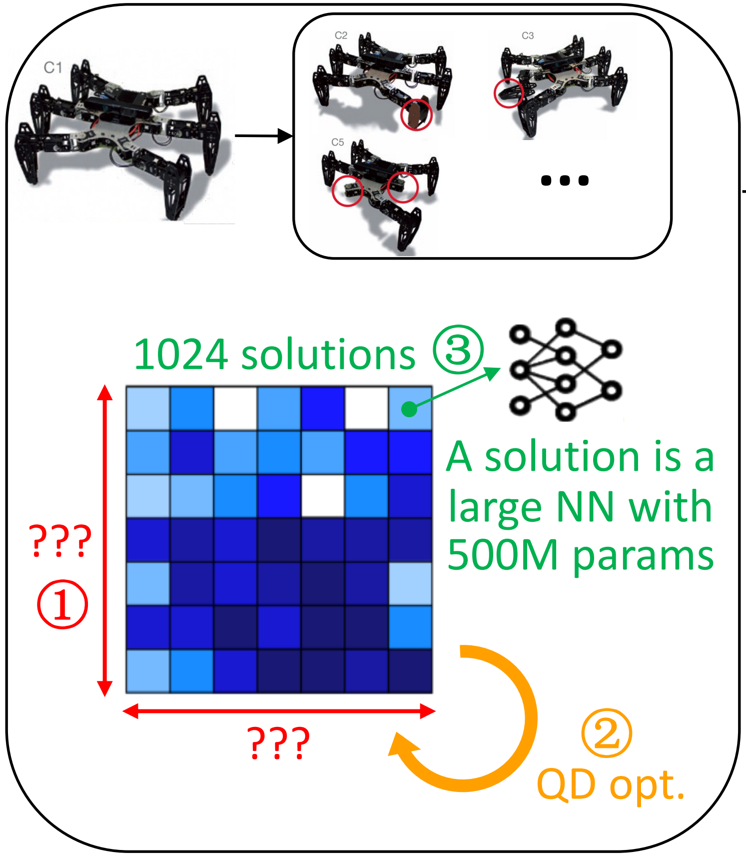
Challenges of Quality-Diversity

- ❑ Can we provide theoretical support for QD?
  - ➢ Prove that QD can be helpful for optimization, i.e., finding a better overall solution
- ❑ How to define the behavior function?
  - ➢ Learn from human feedback
- ❑ **How to improve the sample efficiency?**
  - ➢ **Clustering-based and NSS-based parent selection, cooperative coevolution**
- ❑ How to improve the resource efficiency?
  - ➢ Decomposition and sharing

Challenges of Quality-Diversity

Train diverse policies to adapt to unseen complex environments

Solutions: 1024 diverse policies with 500M parameters
Fitness: Forward distance
Behavior: Learn from human feedback

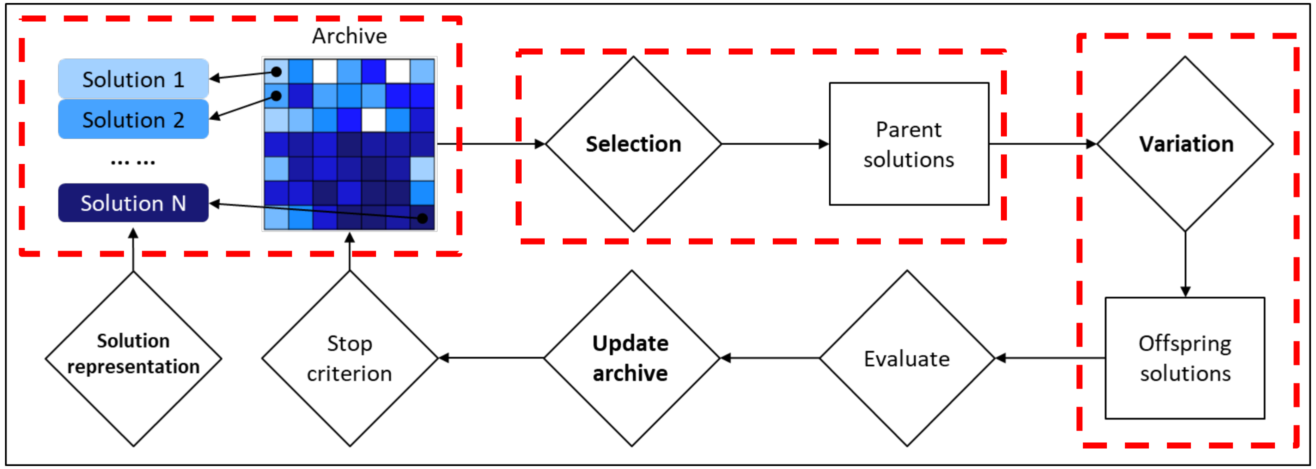


② The optimization of 1024 diverse policies is difficult

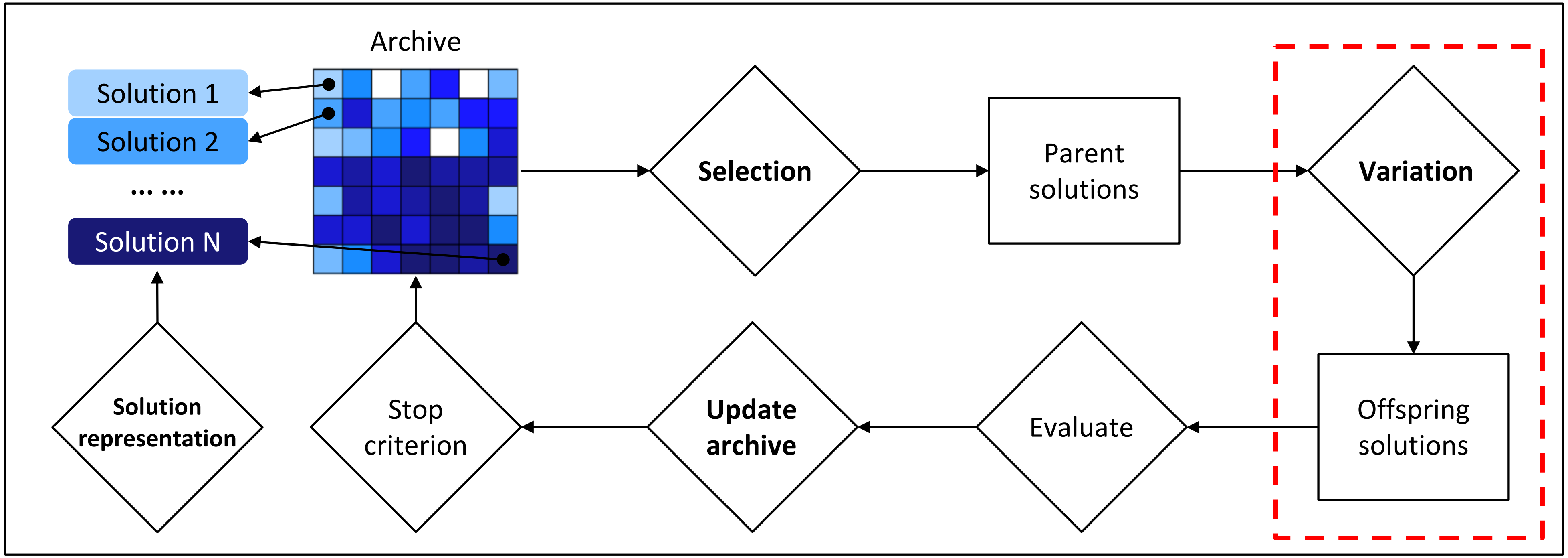
Require sampling many (e.g., 10^9) steps
Low sample efficiency

Challenging tasks

How to improve the efficiency of operators?



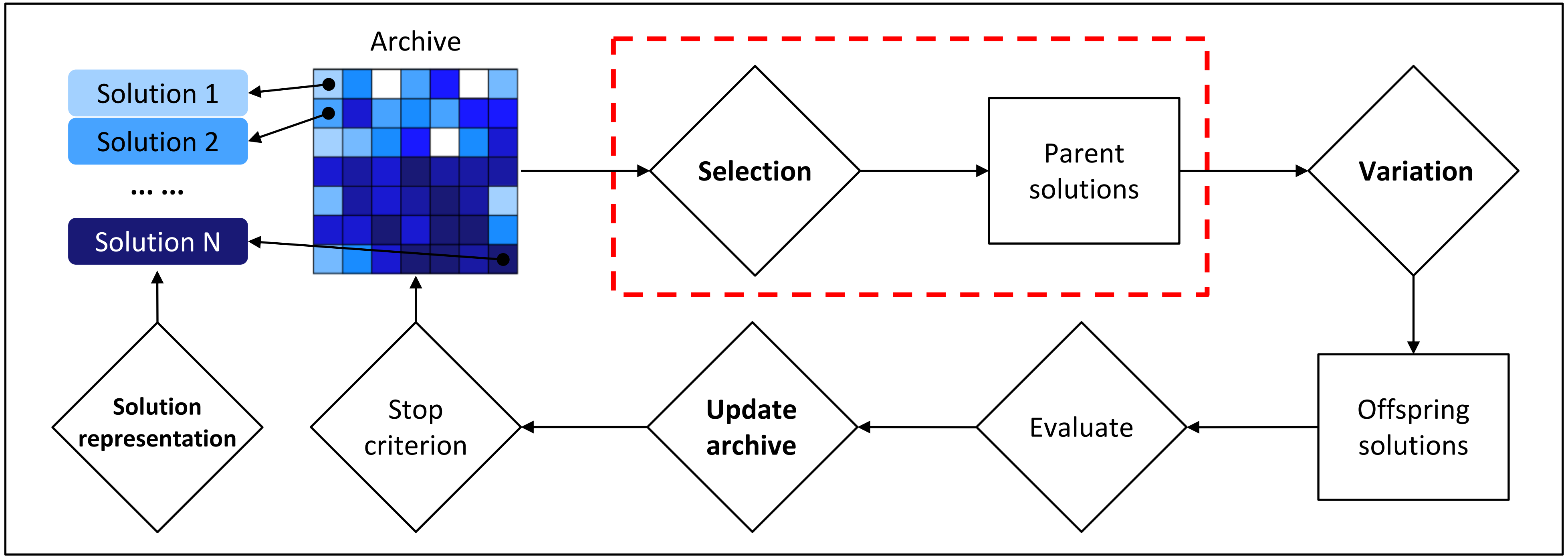
Existing Works on Improving Sample Efficiency



Existing works focus more on the variation process

- PGA-ME uses policy gradient as another operator for variation [Nilsson & Cully, GECCO'21, Best paper award]
- DQD considers the gradients of both fitness and behavior [Fontaine & Nikolaidis, NeurIPS'21 Oral]

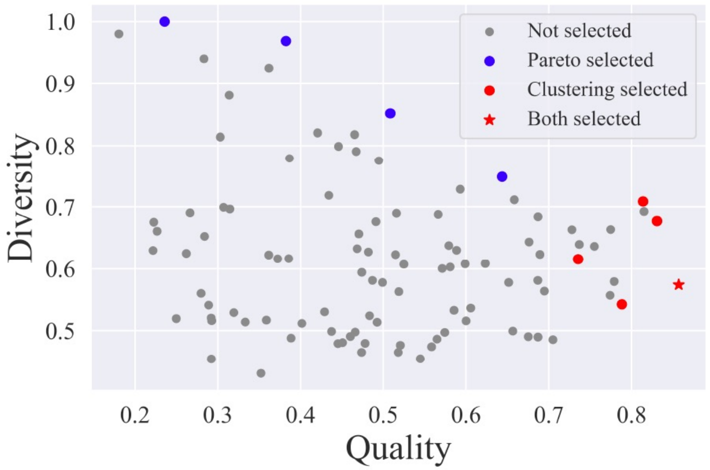
Parent Selection of QD

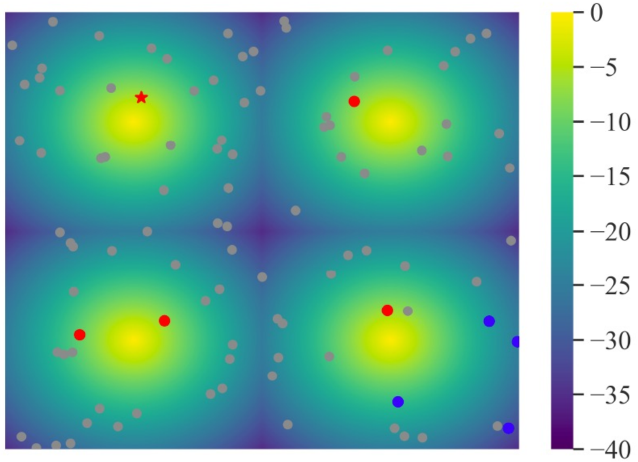


Method	Selection	Reproduction	EAs type	From archive
Vanilla ES	The only parent solution	Quality	$(1, 1)$	×
NSR-ES	Probabilistic selection	Quality and diversity	$(K, 1)$	×
CVT-ES	Uniform selection	Quality and diversity	$(K + K)$	✓
ME-ES	Biased selection	Quality or diversity	$(K + K)$	✓
DvD-ES	All parent solutions	Quality and diversity	(K, K)	×
QD-RL	Pareto-based selection	Quality or diversity	$(K + K)$	✓

Existing parent selection methods are inefficient

Clustering-based Parent Selection





Pareto-based selection [Pierrot et al, arXiv'20] selects blue points: "diverse" in the Pareto space

However, they are not diverse in the behavior space

The red points selected by our method are diverse and have high quality

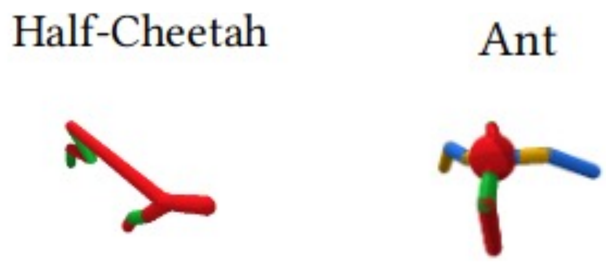
We propose clustering-based selection

- **For Diversity:** Cluster the solutions into different clusters in the behavior space
- **For Quality:** Select one high-quality solution from each cluster

[Wang, Xue, and Qian, ICLR'22]

Experiments on Mujoco Tasks

- **Tasks:** Half-Cheetah and Ant
- **Solution:** Policy parameters
- **Fitness:** Walking (either forward or backward) distance
- **Behavior:** 1 for forward, -1 for backward



Fail to find both modals

forward

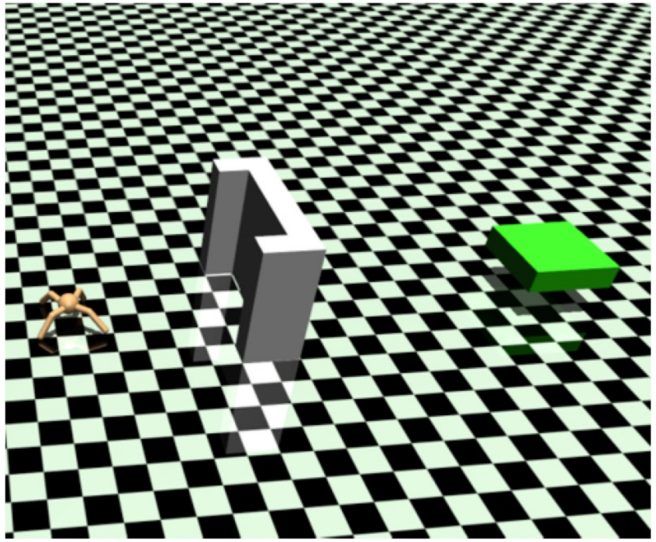
backward

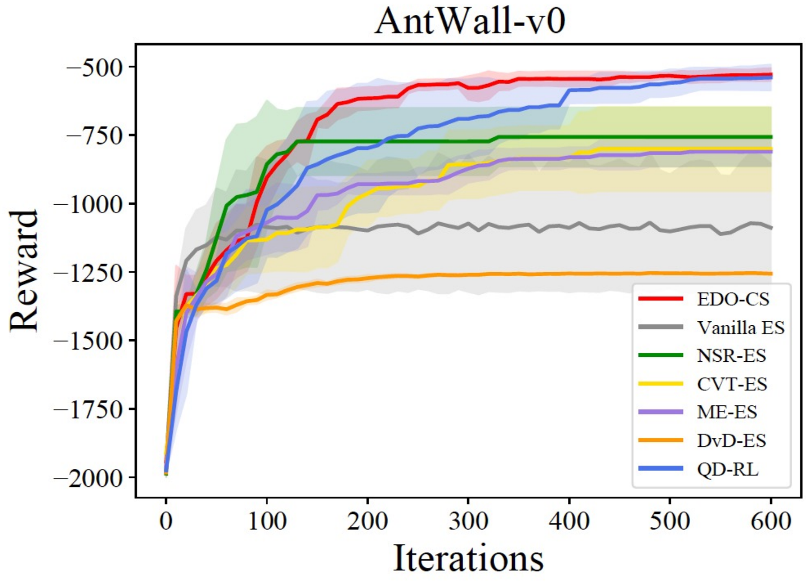
Environment	EDO-CS	QD-RL	ME-ES	DvD-ES	CVT-ES	NSR-ES	Vanilla ES
HalfCheetahFwd	**4284**	2930	2700	-3419	3219	1346	-5543
HalfCheetahBwd	**6548**	6013	5953	6353	4672	5366	3911
AntFwd	**4617**	4291	4316	4507	3856	1737	1911
AntBwd	**4697**	4164	4123	3498	2958	3961	-851
Performance Ranking	**1**	3	3.5	3.75	4.75	5.25	6.75

superior performance under different behaviors

Experiments on Hard Exploration Tasks

- **Task:** Rapidly find policies to circumvent the wall
- **Solution:** Policy parameters
- **Fitness:** Sum of distance to the destination at each step
- **Behavior:** Final location (x, y) of the ant robot





The figure shows the reward of the best policy found by each method

EDO-CS performs better on optimization

[Wang, Xue, and Qian, ICLR'22]

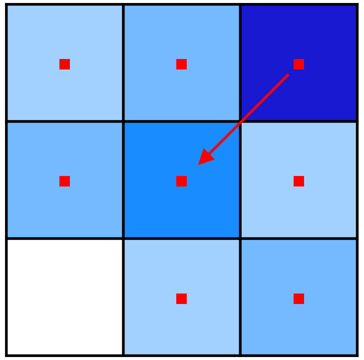
MO-based Parent Selection

A nature method to obtain diversity is by multi-objective optimization [Victor et al, SSCI'21]:

- Consider fitness and behaviors as the objectives to be maximized
- Select solutions by multi-objective optimization

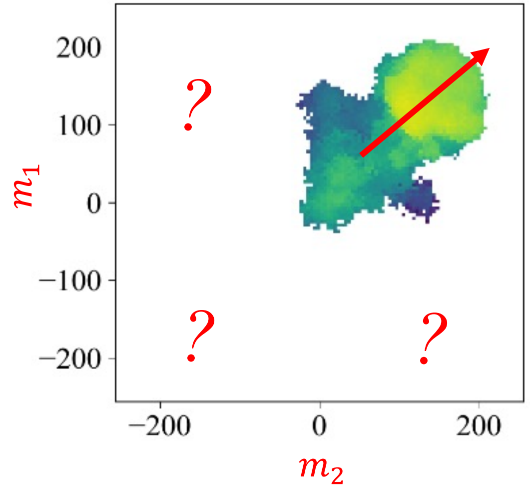
It does not align the goal of QD!

Domination



Color brightness represents the goodness

Evolutionary direction of Pareto dominance



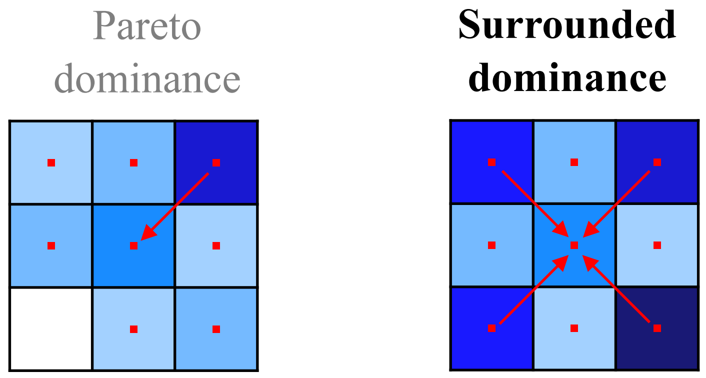
Toward a certain direction of the behavior space, instead of covering the whole behavior space

NSS-based Parent Selection

We propose a new "domination" relationship for QD:

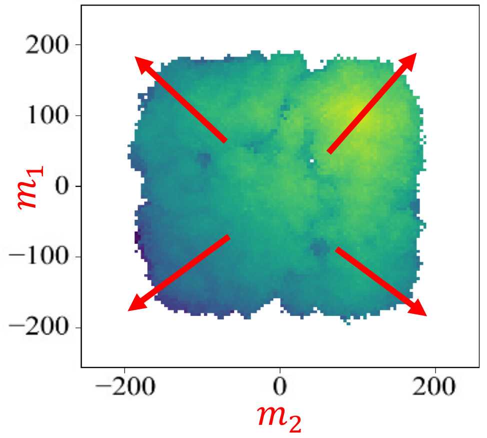
A solution $\boldsymbol{x}$ is **surrounded dominated** if and only if: **for each direction $\boldsymbol{d} \in \{-1, 1\}^k$** in the behavior space, there is another solution $\boldsymbol{x}'$ that

- $\boldsymbol{d} \odot \boldsymbol{m}(\boldsymbol{x}') \succ \boldsymbol{d} \odot \boldsymbol{m}(\boldsymbol{x})$
- $f(\boldsymbol{x}') > f(\boldsymbol{x})$



Surrounded dominance works well as it **considers all directions of the behavior space**

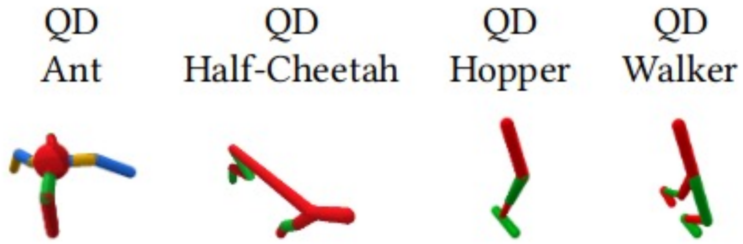
Evolutionary directions of surrounded dominance

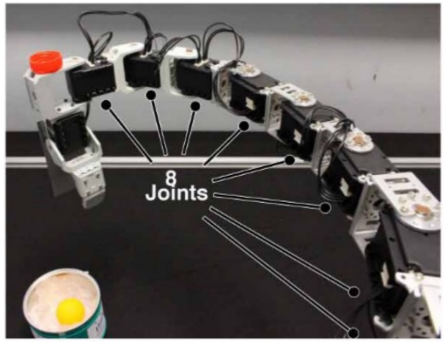


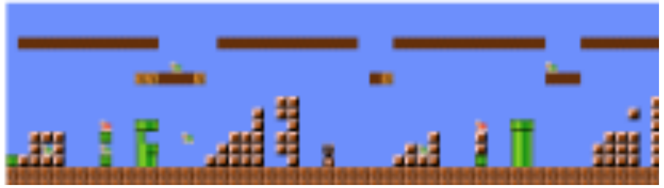
Select the solutions in the top fronts of Non-Surrounded-Dominated Sorting (NSS)

Experiments

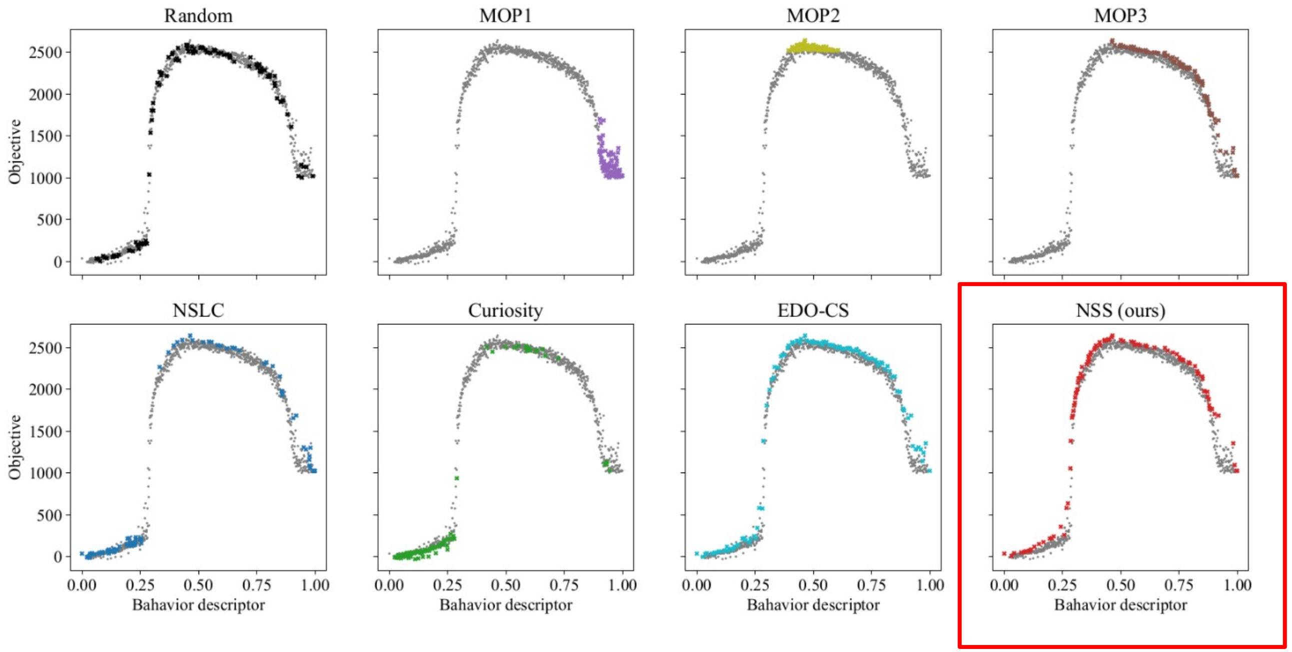
- QDGym
  - Solution: Parameters of policy network
  - Fitness: Mainly determined by forward distance
  - Behavior: Fraction of time each foot touches the ground
- Arm
  - Solution: Angle of each joint
  - Fitness: Negative variance of the joint angles
  - Behavior: Position of the end effector of the arm
- Mario
  - Solution: Latent vector for generating Mario environment
  - Fitness: Completion rate of an agent simulating in the environment
  - Behavior: #tiles of a certain type and #jumps of the agent simulating in the environment

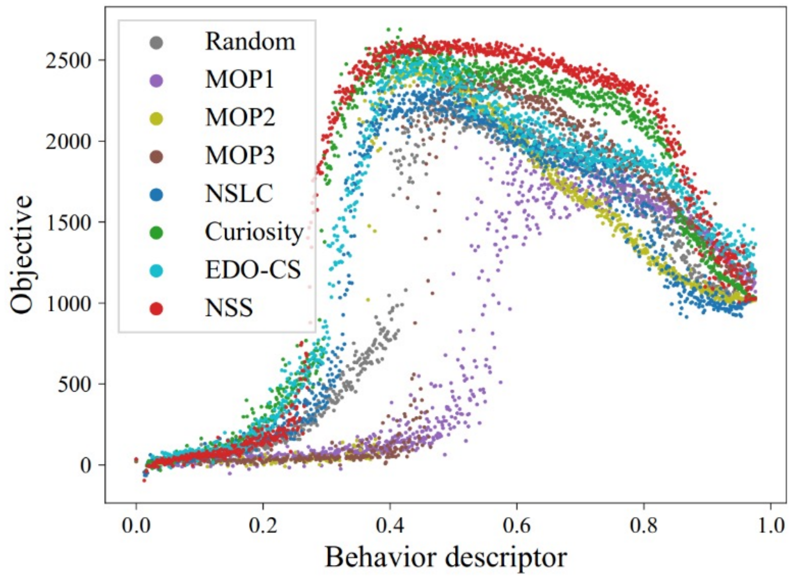






Experiments on QD Hopper



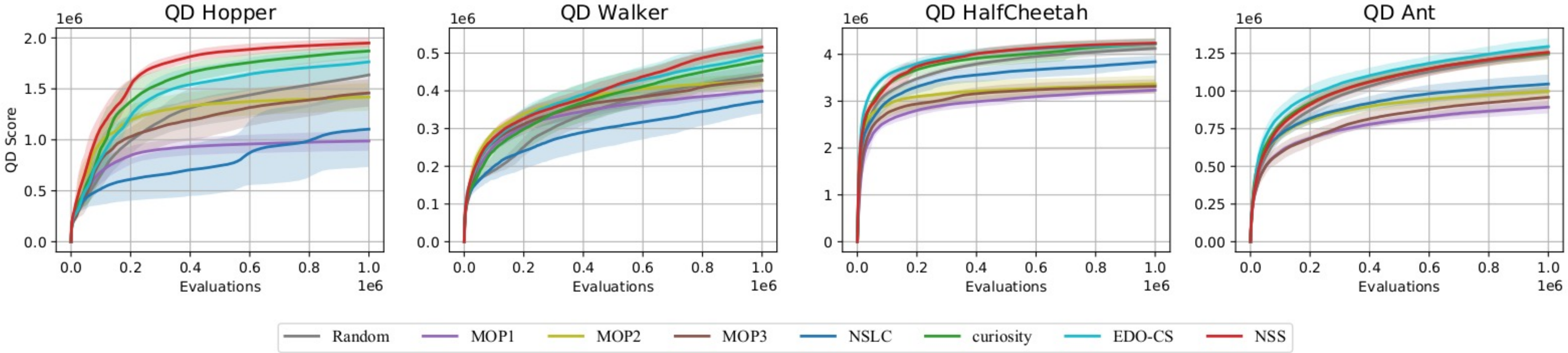


The selected solutions in one generation are diverse and have high quality

The solutions in the final archive are diverse and have high quality

[Wang, Xue, Shang, Qian, Fu, and Fu, IJCAI'23]

Experiments

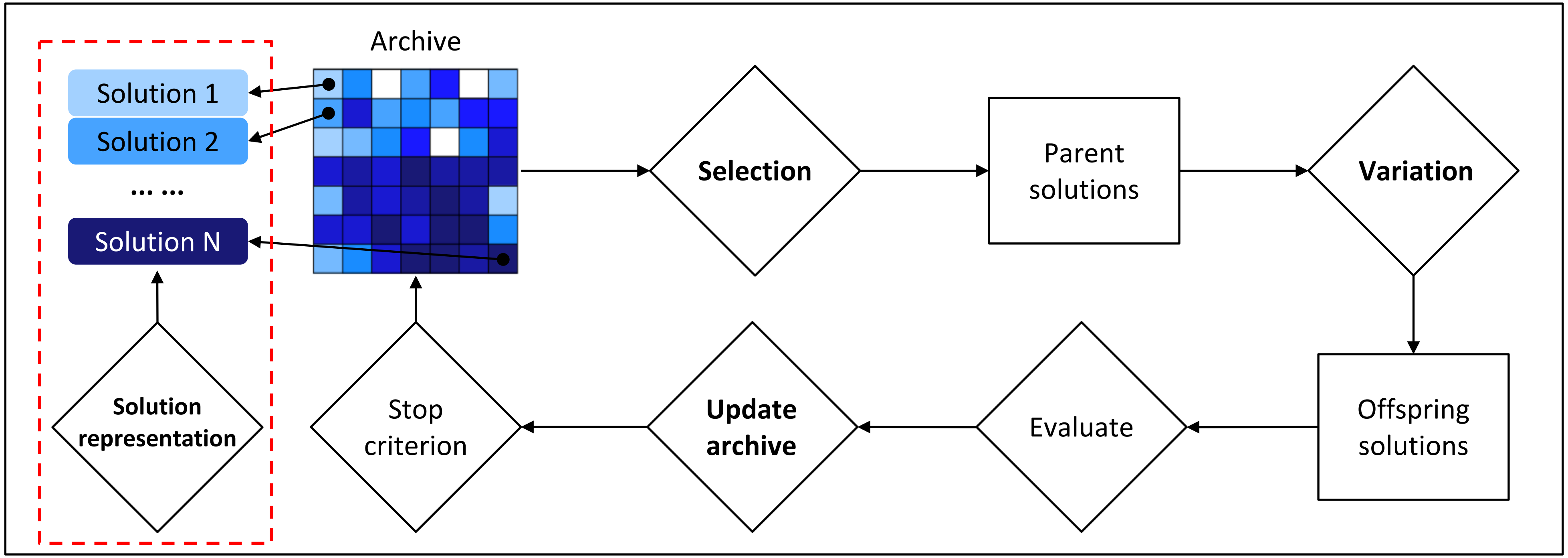


Environment	Random	MOP1	MOP2	MOP3	NSLC	Curiosity	EDO-CS	NSS
QD Hopper	1.25 –	0.88 –	1.26 –	1.17 –	0.78 –	1.53 –	1.44 –	**1.69**
QD Walker	0.34 –	0.34 –	0.37 –	0.35 –	0.29 –	0.36 ≈	0.38 ≈	**0.40**
QD HalfCheetah	3.72 –	2.94 –	3.17 –	3.08 –	3.48 –	3.84 ≈	**3.94** ≈	3.90
QD Ant	1.02 ≈	0.77 –	0.88 –	0.80 –	0.90 –	1.05 ≈	**1.09** ≈	1.04
Arm	18.09 –	17.26 –	9.89 –	20.17 –	**21.77** +	18.30 –	18.59 –	20.52
Mario	98.45 –	41.05 –	38.29 –	40.54 –	112.72 –	**138.65** ≈	108.55 –	134.18
+/–/≈	0/5/1	0/6/0	0/6/0	0/6/0	1/5/0	0/2/4	0/3/3	
Average Rank	5.00	7.00	5.83	5.83	5.00	2.83	2.50	**1.83**

NSS achieves the highest average rank of QD-Score AUC

[Wang, Xue, Shang, Qian, Fu, and Fu, IJCAI'23]

Solution Representation of QD



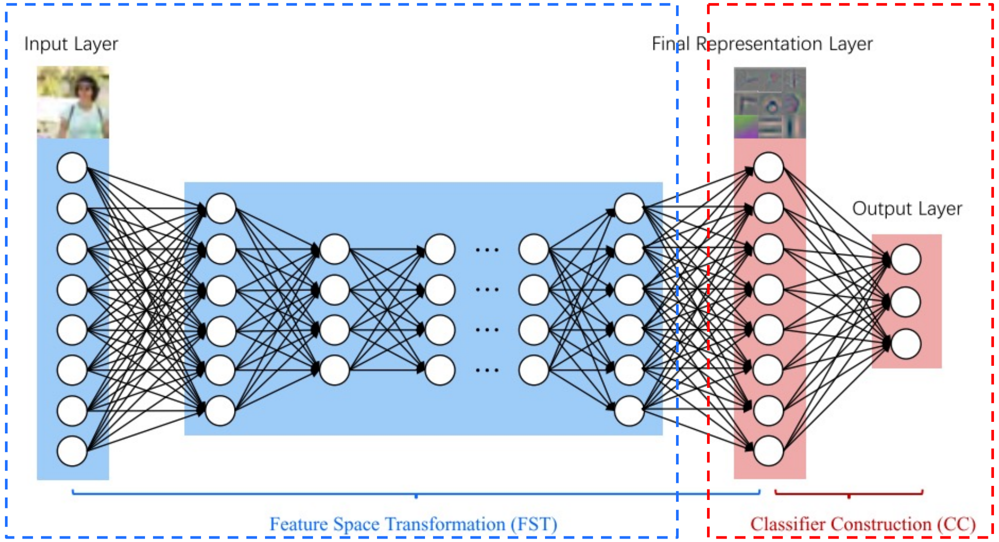
For the solution representation of QD

- QD maintains a large number of solutions, each is a network with a large number of parameters
- The optimization space is excessively large

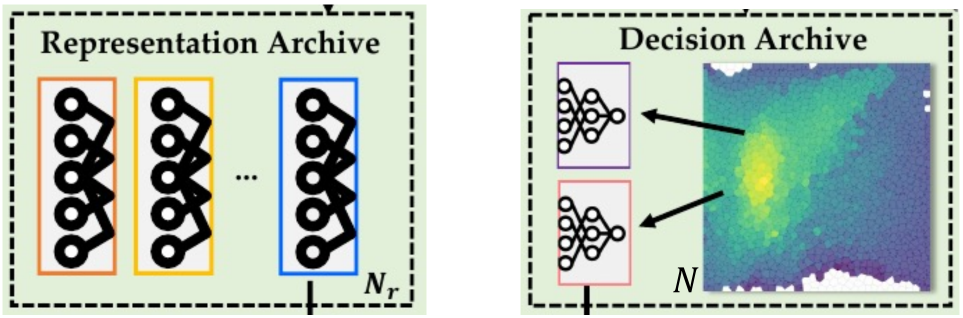
QD by Cooperative Coevolution

Observation: Different layers of a policy network have different functions

- To reduce the difficulty of optimization
  - ➢ decompose the policy network into two parts by layers
  - ➢ maintain archives for the two parts respectively
  - ➢ optimize them by cooperative coevolution
- To further reduce the optimization space
  - ➢ reduce the number of representation parts
  - ➢ share representation knowledge



[Zhou, SCIS'21] Representation part Decision part



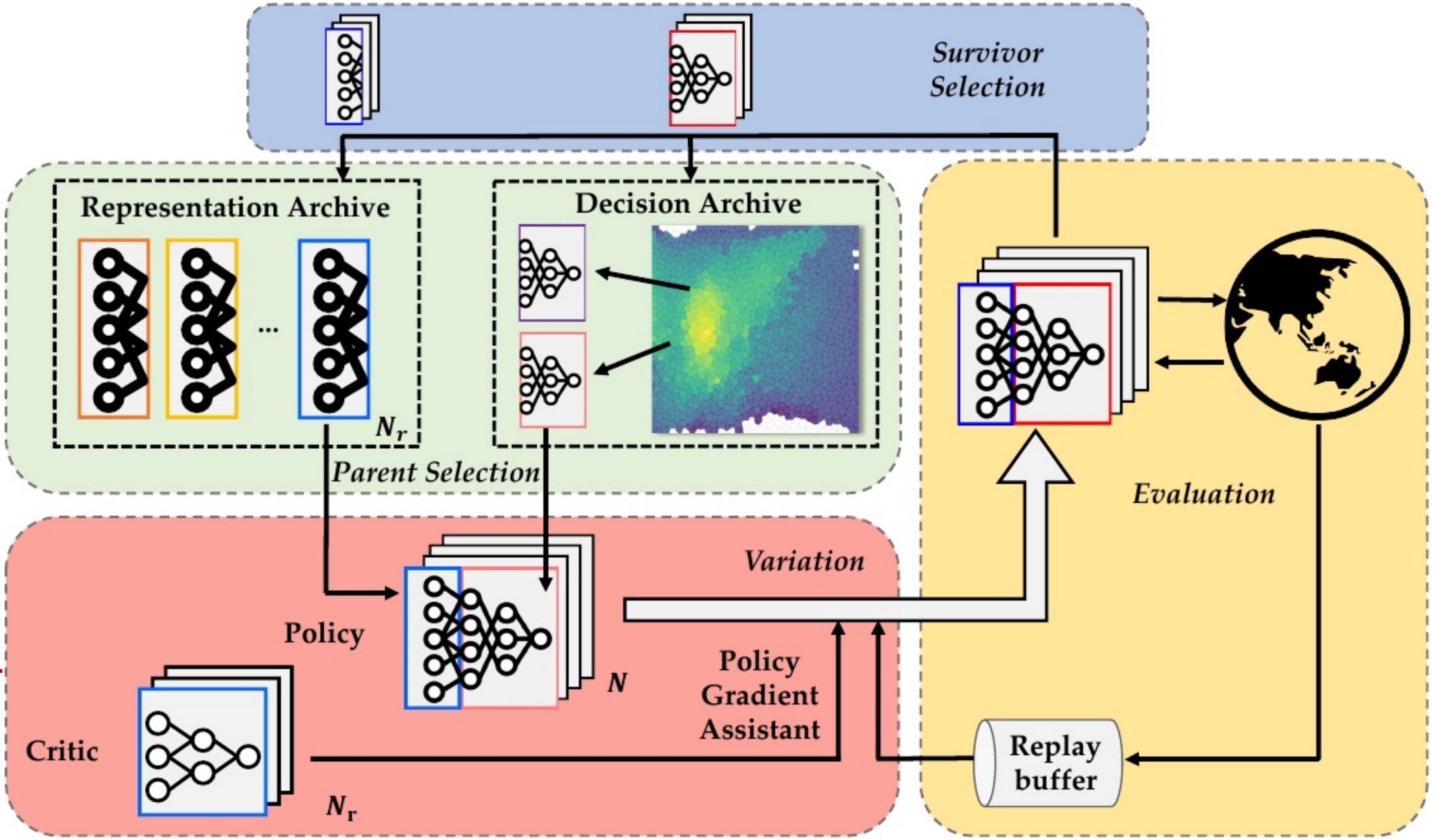
$N_r \ll N$

[Xue, Wang, Li, Li, Hao, and Qian, ICLR'24 Spotlight]

QD by Cooperative Coevolution

Survivor selection for each part

Maintain an archive for each part



Evaluation

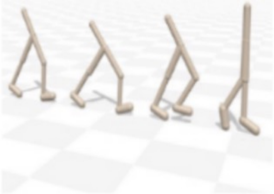
Select parents from each population, combine and variate the solutions

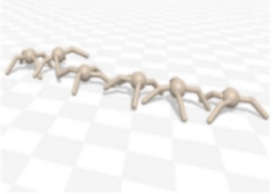
[Xue, Wang, Li, Li, Hao, and Qian, ICLR'24 Spotlight]

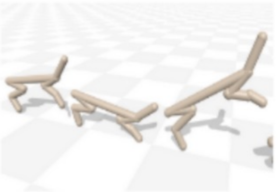
Experiments

On 8 QDax tasks and 1 Atari task using policy parameters as solutions

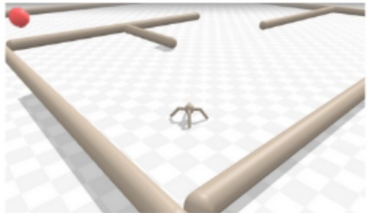
- Uni (Omni) tasks:
  - Fitness: Weighted sum of forward distance and energy cost
  - Behavior function of Uni: Fraction of time each foot touches the ground
  - Behavior function of Omni: Final position of the robot
- Maze tasks:
  - Fitness: Sum of negative distance to the target position
  - Behavior function: Final position of the robot
- Atari Pong:
  - Fitness: Points winning in the game
  - Behavior function: Frequency of movement

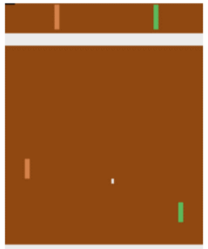

Walker2D


Ant


HalfCheetah


Point Maze

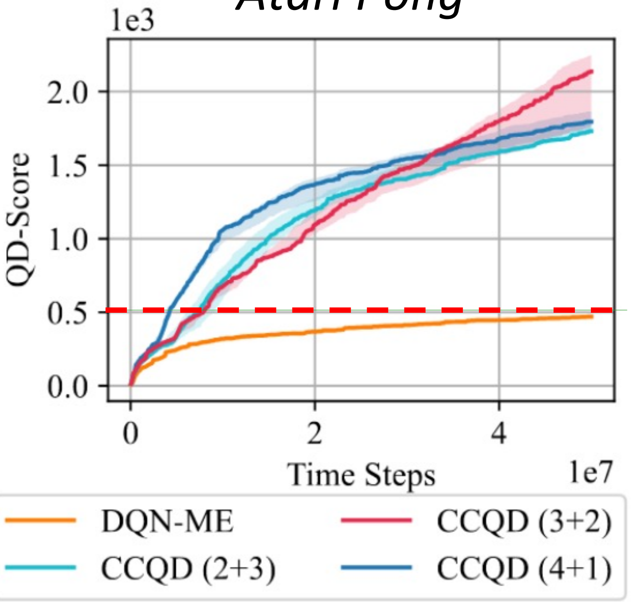

Ant Maze


Pong

Experiments

Environment	ME	QD-PG	PGA-ME	OMG-MEGA	PBT-ME	CCQD
Hopper Uni	84.17 –	75.20 –	93.25 –	91.47 –	81.32 –	**96.75**
Walker2D Uni	102.73 –	103.36 –	109.56 –	110.23 –	85.20 –	**116.83**
HalfCheetah Uni	343.79 –	323.44 –	388.24 –	392.61 –	425.16 ≈	**432.83**
Ant Uni	121.16 –	131.10 –	131.90 –	135.98 ≈	121.89 –	**141.27**
Humanoid Uni	119.36 –	125.09 –	116.36 –	117.43 –	97.61 –	**132.51**
Humanoid Omni	0.90 –	1.45 –	1.40 –	1.07 –	1.22 –	**2.65**
Point Maze	43.90 –	42.74 –	35.09 –	34.63 –	35.01 –	**52.73**
Ant Maze	105.90 –	**164.94** ≈	141.64 –	146.46 –	132.47 –	157.03
+/ – / ≈	0/8/0	0/7/1	0/8/0	0/7/1	0/7/1	/
Average Rank	4.62	3.50	3.50	3.50	4.75	**1.12**

Atari Pong

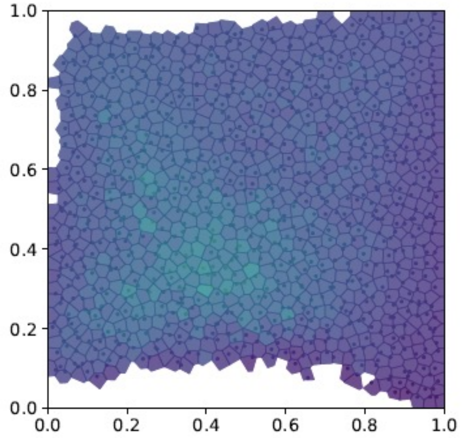


CCQD achieves the highest average rank of QD-Score AUC with the same number of samples

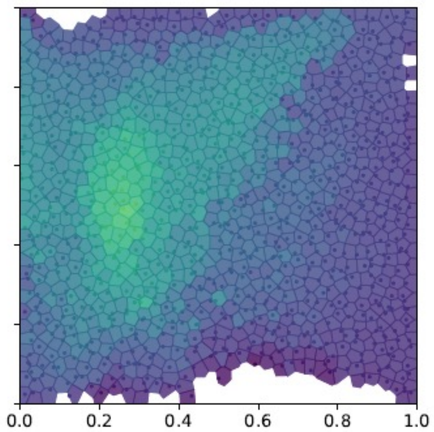
On the challenging task *Atari Pong*, CCQD uses only less than 20% samples to obtain the same QD-Score

[Xue, Wang, Li, Li, Hao, and Qian, ICLR'24 Spotlight]

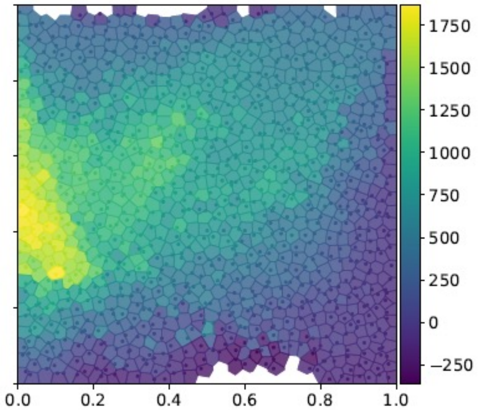
Experiments



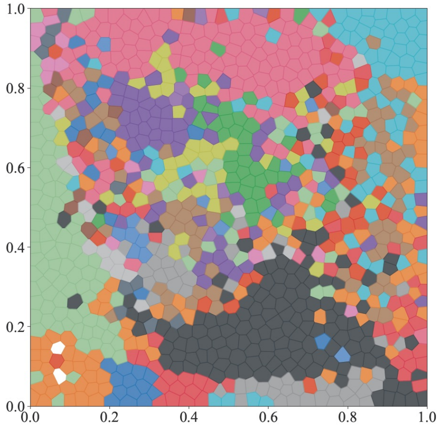
(a) ME



(b) PGA-ME



(c) CCQD



(d) CCQD

(a)-(c): Visualization of the final archives of different methods

CCQD has the best archive

(d): Different colors denote different representation parts

Different representation parts can discover different behaviors

[Xue, Wang, Li, Li, Hao, and Qian, ICLR'24 Spotlight]

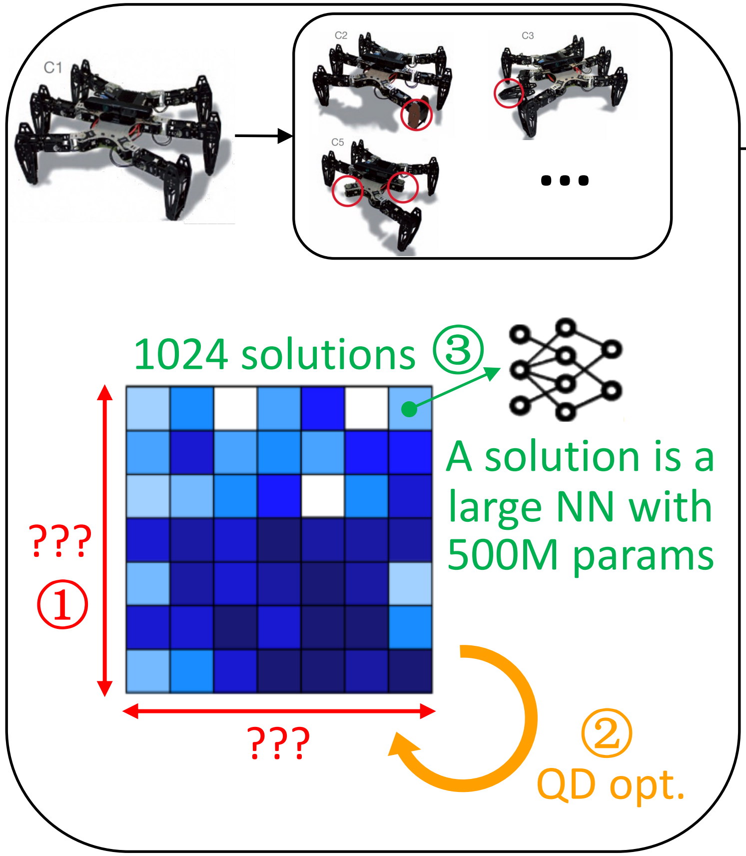
Challenges of Quality-Diversity

- Can we provide theoretical support for QD?
  - Prove that QD can be helpful for optimization, i.e., finding a better overall solution
- How to define the behavior function?
  - Learn from human feedback
- How to improve the sample efficiency?
  - Clustering-based and NSS-based parent selection, cooperative coevolution
- **How to improve the resource efficiency?**
  - **Decomposition and sharing**

Challenges of Quality-Diversity

Train diverse policies to adapt to unseen complex environments

Solutions: 1024 diverse policies with 500M parameters
Fitness: Forward distance
Behavior: Learn from human feedback

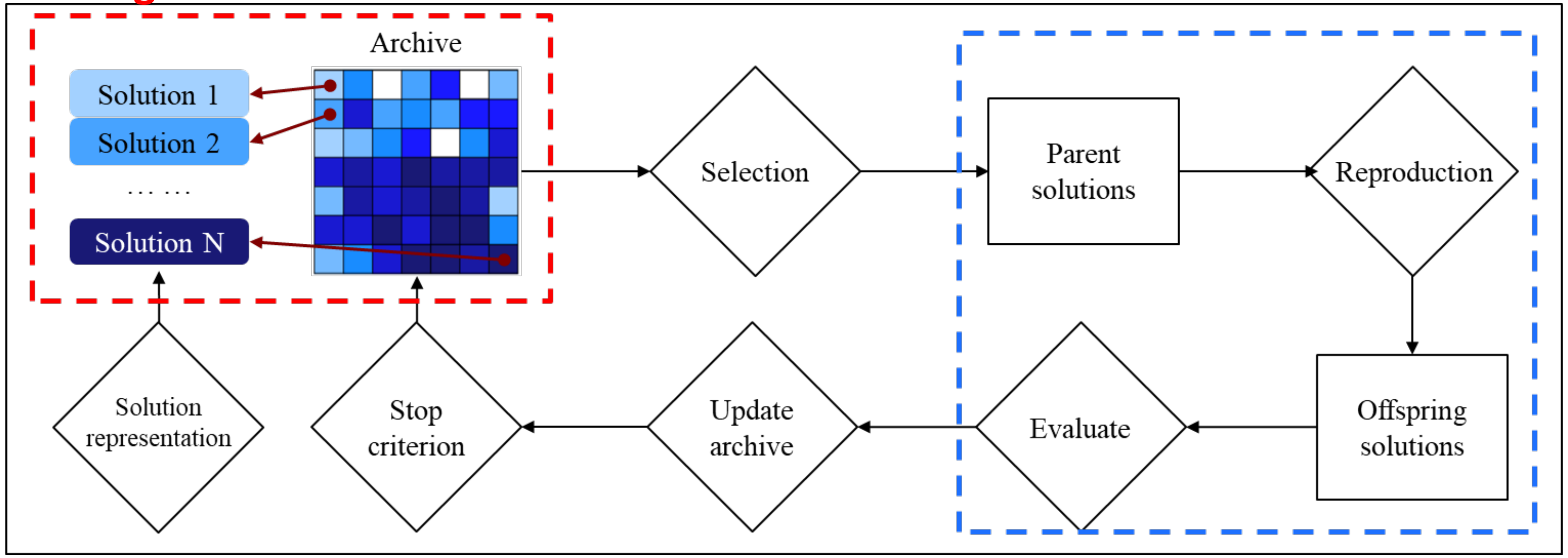


③ → Require $\geq$ 400G GPU memory and $\geq$ 2000G RAM → Unable to train on typical computational servers

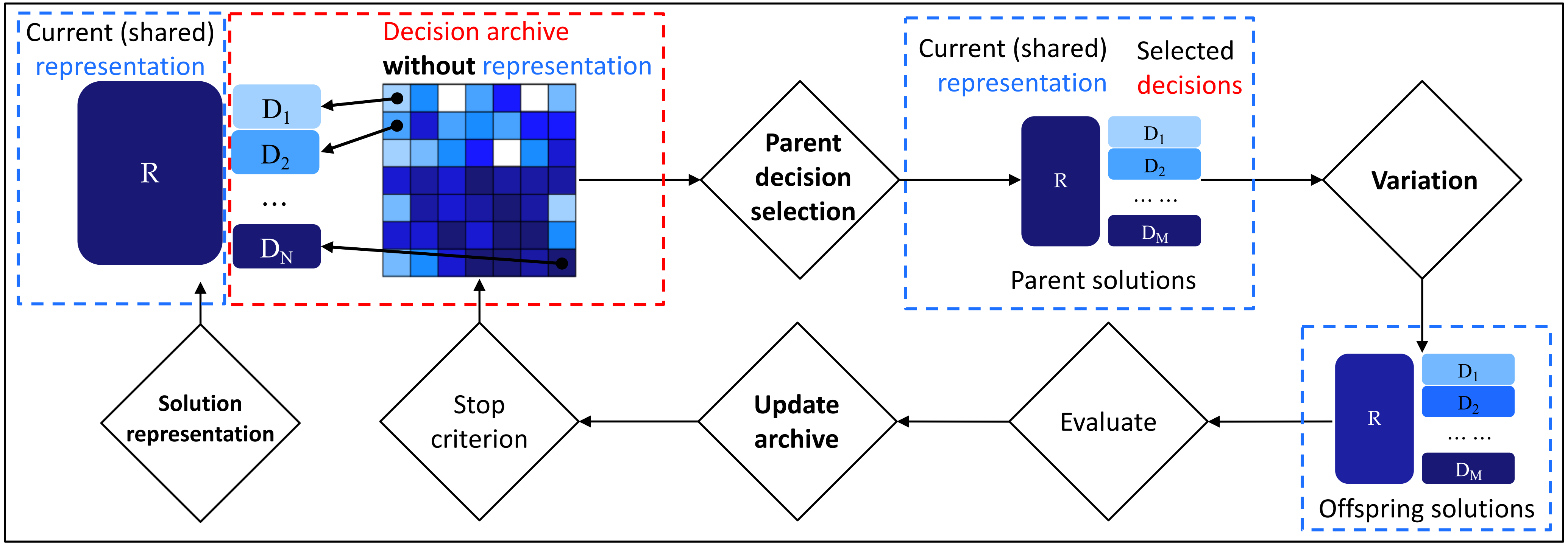
Low resource efficiency

Large archive in RAM

Train & Evaluate on GPU



Resource-efficient QD



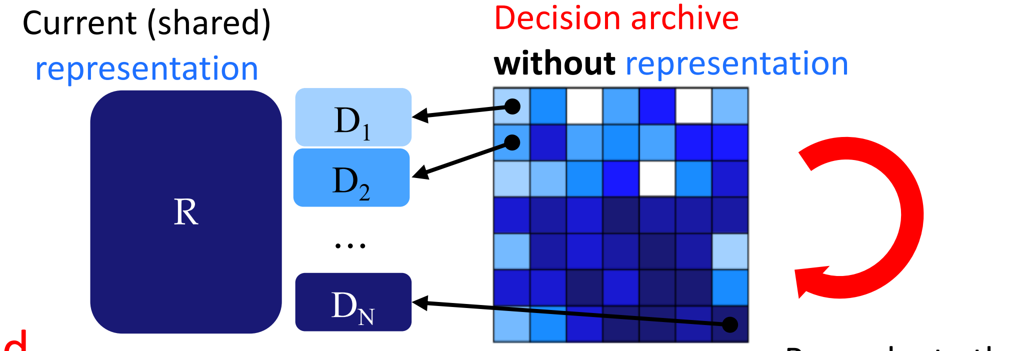
- Maintain a single shared representation part only -> Reduce GPU memory overhead
- Do not save representation parts in decision archive -> Reduce RAM overhead
  - ➢ But lead to the **mismatch problem** between the shared representation part and decision parts

[Wang, Xue, Guan, and Qian, ICML'24]

Resource-efficient QD

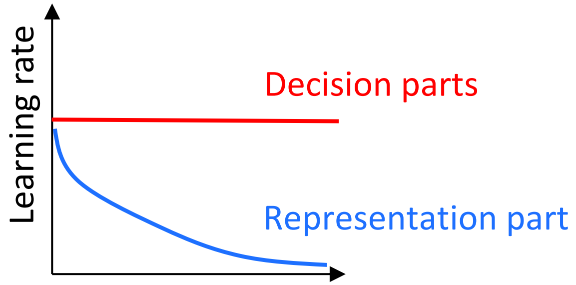
Solve the **mismatch problem** between the shared representation part and decision parts:

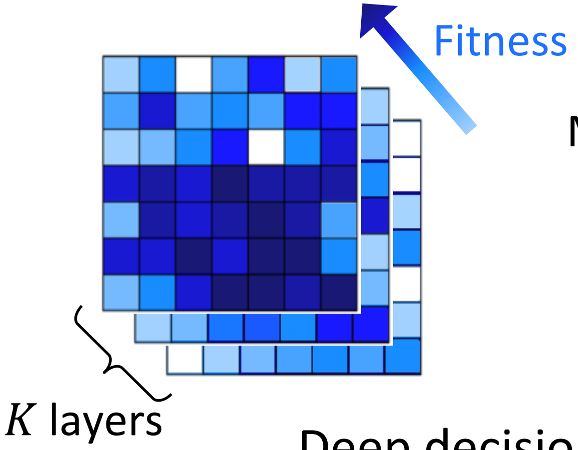
- **Periodic re-evaluation**: Re-evaluate the decision parts and add them back by survivor selection periodically
- **Deep decision archive**: Maintain K decision parts instead of one in each cell of the decision archive to improve robustness
- **Learning rate decay**: Decay the learning rate of the representation part to improve its stability



Periodic re-evaluation

Learning rate decay





Maintain K decision parts in each cell

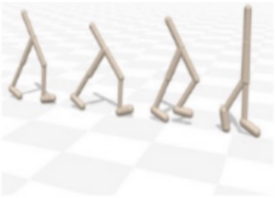
The top layer has the best fitness

Deep decision archive

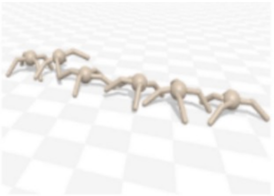
Experiments

On **8 QDax tasks** and **2 Atari tasks** using policy parameters as solutions

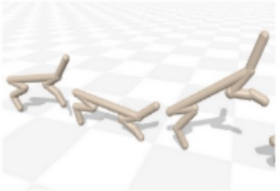
- Uni tasks:
  - Fitness: Mainly determined by forward distance of the robot
  - Behavior: Fraction of time each foot touches the ground
- Maze and Trap tasks:
  - Fitness: Sum of negative distance to the target position
  - Behavior: Final position of the robot
- Atari tasks:
  - Fitness: Points winning in the game
  - Behavior:
    - Pong: Frequency of movement
    - Boxing: Frequency of movement and punches



Walker2D Uni

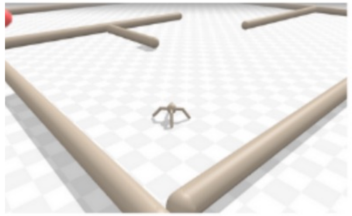


Ant Uni

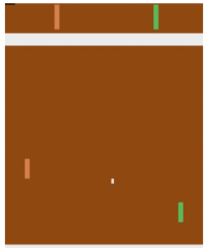


HalfCheetah Uni

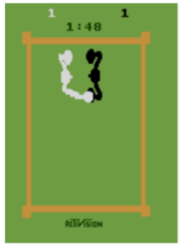
Point-Maze



Ant-Maze



Pong

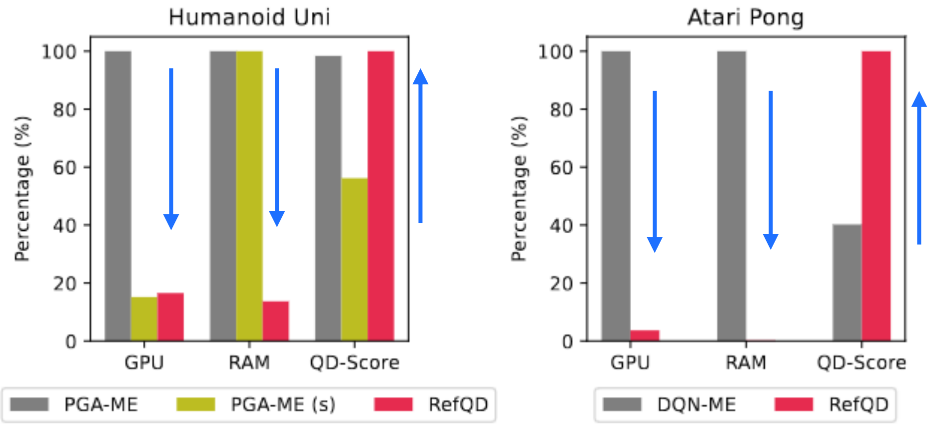


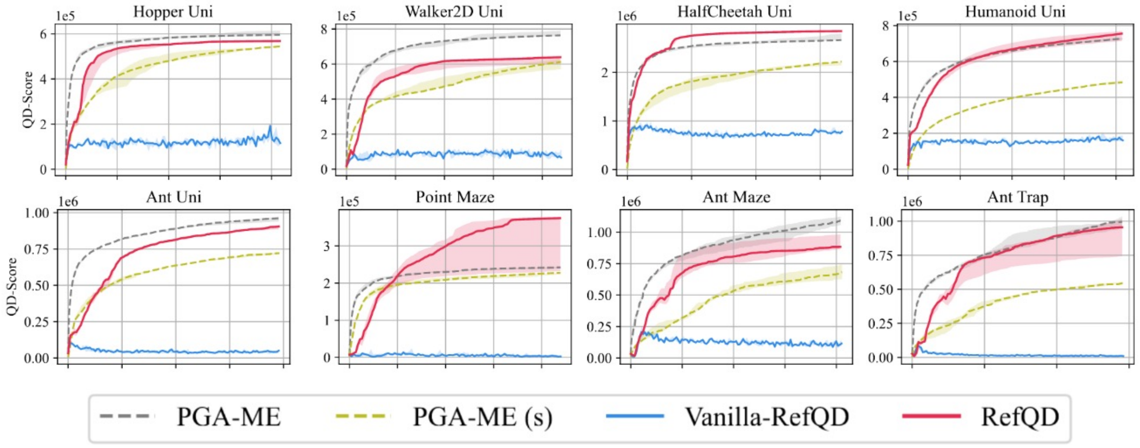
Boxing

Experiments

Compared methods:

- PGA-ME: The SOTA QD method using unlimited resources
- PGA-ME (s): PGA-ME with a small number of offsprings, using less GPU memory but the same RAM
- Vanilla-RefQD: The vanilla version of RefQD, which does not use the strategies for solving mismatch issue





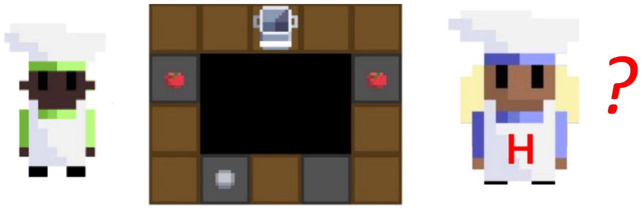
RefQD achieves comparable QD-Score with significantly fewer resources

Challenges of Quality-Diversity

- **Can we provide theoretical support for QD?**
  - Prove that QD can be helpful for optimization, i.e., finding a better overall solution
- **How to define the behavior function?**
  - Learn from human feedback
- **How to improve the sample efficiency?**
  - Clustering-based and NSS-based parent selection, cooperative coevolution
- **How to improve the resource efficiency?**
  - Decomposition and sharing

Application: Human-AI Coordination

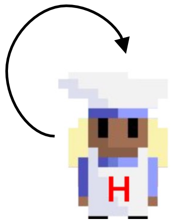
Zero-shot coordination (ZSC) aims at training agents that can coordinate well with unseen human partners.



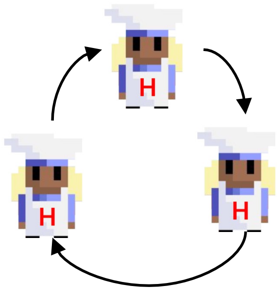
How to achieve that?

Train with diverse partners

Self-play (SP)

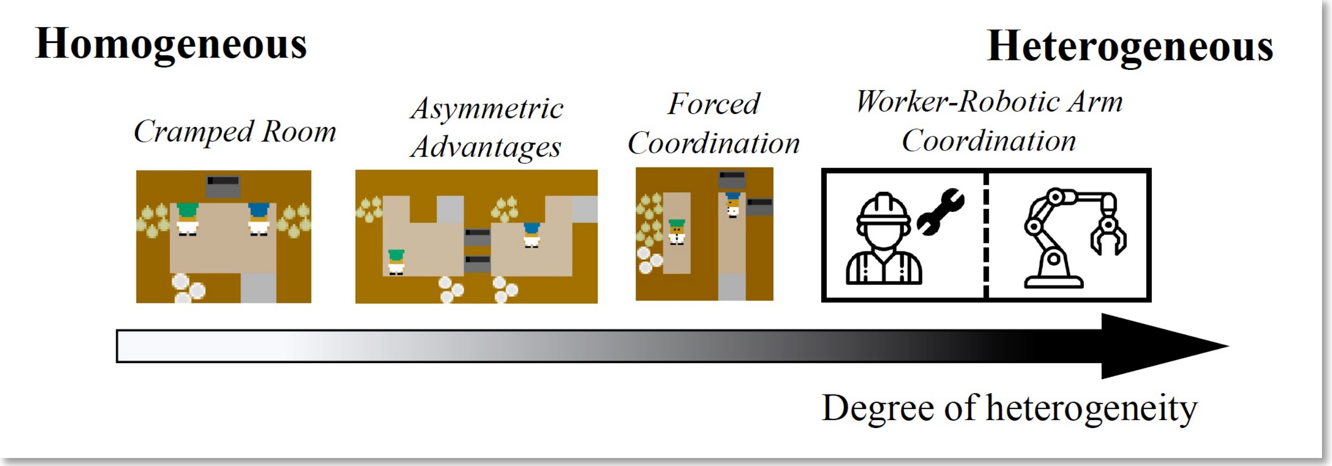


Population-play (PP)



Heterogeneous ZSC

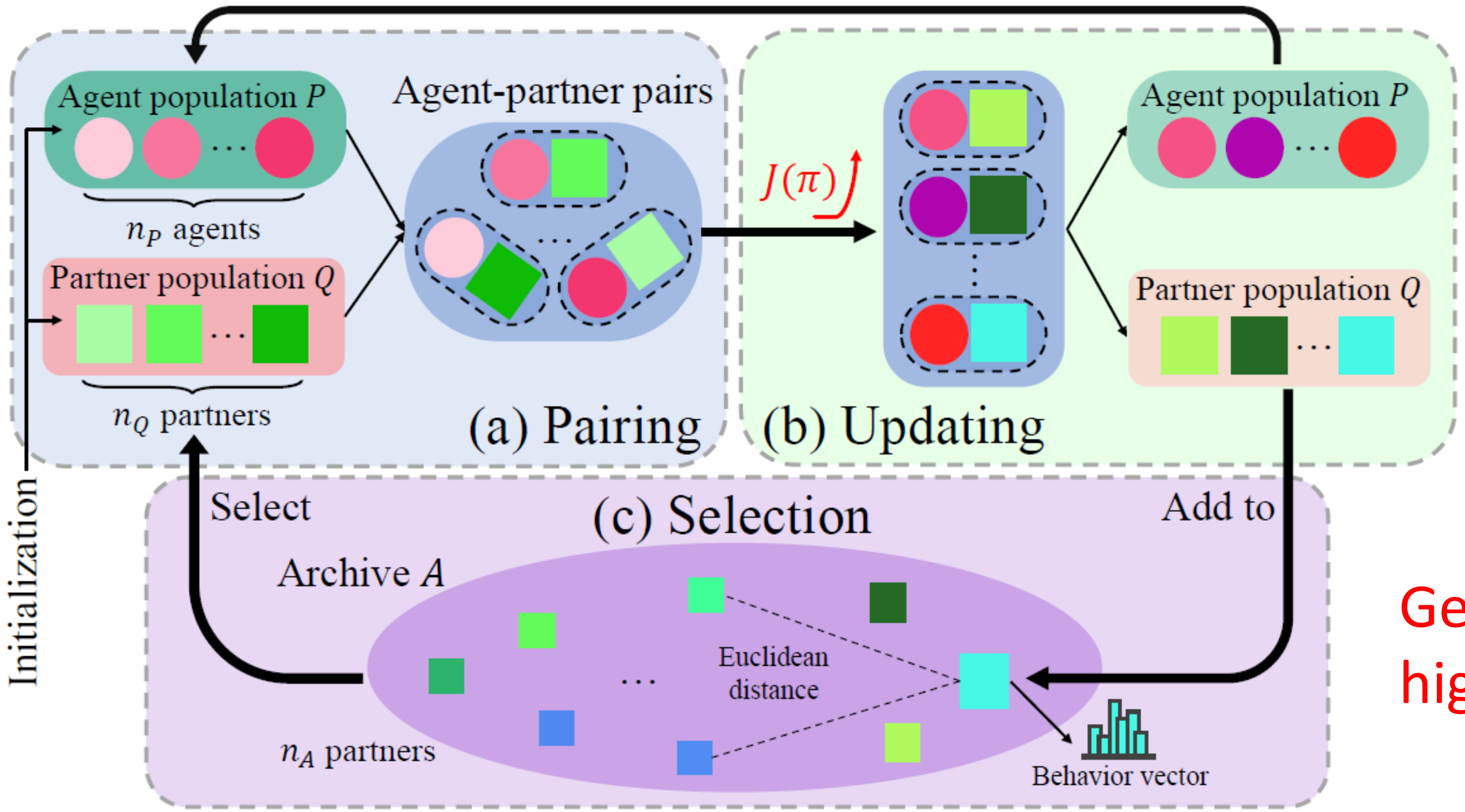
- In many real-world applications, human and AI are heterogenous, i.e., human and AI have different action spaces.
- Traditional SP and PP (homogeneous training approach) do not work!



[Xue, Wang, Guan, Yuan, Fu, Fu, Qian, and Yu. Under Review]

Application: Human-AI Coordination

We propose a heterogeneous framework based on cooperative coevolution



Generate a diverse archive of high-performing partners

[Xue, Wang, Guan, Yuan, Fu, Fu, Qian, and Yu. Under Review]

Application: Human-AI Coordination

LAMDA
Learning And Mining from DatA
http://www.lamda.nju.edu.cn

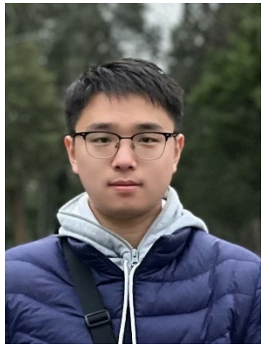
Layout	Partner	SP	PP	TrajeDi	FCP	MEP	MAZE
CR	Random	45.6 ± 12.0 −	61.8 ± 12.9 −	82.0 ± 13.9 −	121.4 ± 11.0 ≈	129.8 ± 7.1 ≈	**130.4 ± 19.0**
	Self-Play	89.6 ± 15.5 −	116.8 ± 23.1 ≈	**143.4 ± 19.2** ≈	141.6 ± 16.6 ≈	91.0 ± 14.7 −	122.8 ± 11.6
	MAZE	186.4 ± 15.6 ≈	199.0 ± 11.5 ≈	190.6 ± 23.6 ≈	**201.2 ± 4.0** ≈	154.6 ± 22.3 −	193.6 ± 19.5
	Human Proxy	97.6 ± 13.3 ≈	90.0 ± 14.6 ≈	100.5 ± 22.6 ≈	118.0 ± 34.1 ≈	106.4 ± 18.6 ≈	**121.0 ± 28.0**
CR-2	Random	3.8 ± 2.1 −	5.0 ± 1.5 −	5.5 ± 0.7 −	6.4 ± 1.1 ≈	6.5 ± 0.7 ≈	**7.0 ± 0.7**
	Self-Play	40.4 ± 9.5 −	36.8 ± 6.4 −	87.6 ± 9.0 ≈	**121.6 ± 12.9** +	85.8 ± 7.5 ≈	97.5 ± 14.6
	MAZE	100.0 ± 14.4 −	143.7 ± 17.1 −	153.8 ± 16.5 −	140.4 ± 15.5 −	129.8 ± 14.9 −	**183.7 ± 11.9**
	Human Proxy	61.4 ± 10.3 −	73.6 ± 10.3 −	81.0 ± 13.3 −	101.2 ± 15.5 ≈	90.2 ± 10.0 −	**118.6 ± 13.2**
AA	Random	30.6 ± 7.6 −	36.0 ± 9.1 −	49.3 ± 11.9 −	50.4 ± 7.4 −	63.5 ± 12.5 ≈	**74.4 ± 15.5**
	Self-Play	172.6 ± 10.4 +	**204.8 ± 10.8** +	180.5 ± 15.5 +	171.0 ± 11.5 +	190.3 ± 10.1 +	142.6 ± 20.3
	MAZE	213.8 ± 17.0 −	223.0 ± 10.1 −	247.0 ± 27.1 −	236.0 ± 12.9 −	120.0 ± 12.4 −	**315.5 ± 10.1**
	Human Proxy	38.5 ± 10.0 −	43.9 ± 2.9 −	50.2 ± 6.0 −	38.4 ± 4.2 −	92.0 ± 29.5 ≈	**112.0 ± 19.1**
AA-2	Random	28.0 ± 6.7 −	29.8 ± 7.0 −	45.5 ± 10.1 ≈	43.5 ± 10.5 ≈	**58.4 ± 14.8** ≈	56.5 ± 9.3
	Self-Play	52.4 ± 10.4 −	79.6 ± 19.4 −	100.8 ± 12.5 −	121.2 ± 15.5 ≈	108.0 ± 29.9 ≈	**130.6 ± 18.5**
	MAZE	132.6 ± 29.7 −	152.8 ± 24.6 −	170.5 ± 11.1 −	156.0 ± 15.3 −	124.6 ± 29.3 −	**381.4 ± 13.7**
	Human Proxy	39.4 ± 6.2 −	33.8 ± 9.3 −	48.0 ± 6.9 −	55.8 ± 15.2 −	79.0 ± 11.2 −	**111.5 ± 13.7**
FC	Random	3.3 ± 2.2 −	3.0 ± 0.6 −	6.0 ± 0.7 −	5.1 ± 0.7 −	6.0 ± 1.0 −	**7.3 ± 0.4**
	Self-Play	74.6 ± 17.6 −	76.8 ± 25.2 −	80.8 ± 22.4 −	79.5 ± 22.4 −	82.3 ± 22.6 −	**147.0 ± 27.1**
	MAZE	96.5 ± 16.0 −	92.6 ± 11.6 −	115.6 ± 25.7 −	119.8 ± 10.9 −	95.2 ± 15.7 −	**164.0 ± 27.0**
	Human Proxy	21.3 ± 2.4 ≈	28.1 ± 13.3 ≈	25.4 ± 6.8 ≈	26.9 ± 7.4 ≈	**28.6 ± 5.3** ≈	25.0 ± 5.1
+/ − / ≈		1/16/3	1/15/4	1/12/6	2/10/8	1/10/9	
Average Rank		5.43	4.42	3.18	3.00	3.22	**1.75**

Our method achieves the best average performance

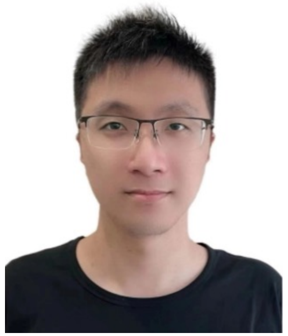
[Xue, Wang, Guan, Yuan, Fu, Fu, Qian, and Yu. Under Review]

Collaborators

Learning And Mining from DatA
http://www.lamda.nju.edu.cn



Ke Xue (Ph.D. 22)



Ren-Jian Wang (Ph.D. 24)

Thank you!

Main References

[1] C. Qian, K. Xue, and R.-J. Wang. Quality-diversity algorithms can provably be helpful for optimization. IJCAI'24.

[2] R.-J. Wang, K. Xue, Y. Wang, P. Yang, H. Fu, Q. Fu, and C. Qian. Diversity from human feedback. NeurIPS'23 ALOE Workshop.

[3] Y. Wang, K. Xue, and C. Qian. Evolutionary diversity optimization with clustering-based selection for reinforcement learning. ICLR'22.

[4] R.-J. Wang, K. Xue, H. Shang, C. Qian, H. Fu, and Q. Fu. Multi-objective optimization-based selection for quality-diversity by non-surrounded-dominated sorting. IJCAI'23.

[5] K. Xue, R.-J. Wang, P. Li, D. Li, J. Hao, and C. Qian. Sample-efficient quality-diversity by cooperative coevolution. ICLR'24.

[6] R.-J. Wang, K. Xue, C. Guan, and C. Qian. Quality-diversity with limited resources. ICML'24.

[1] Provide theoretical support for QD

[2] Make QD easier to use

[3-6] Make QD more efficient