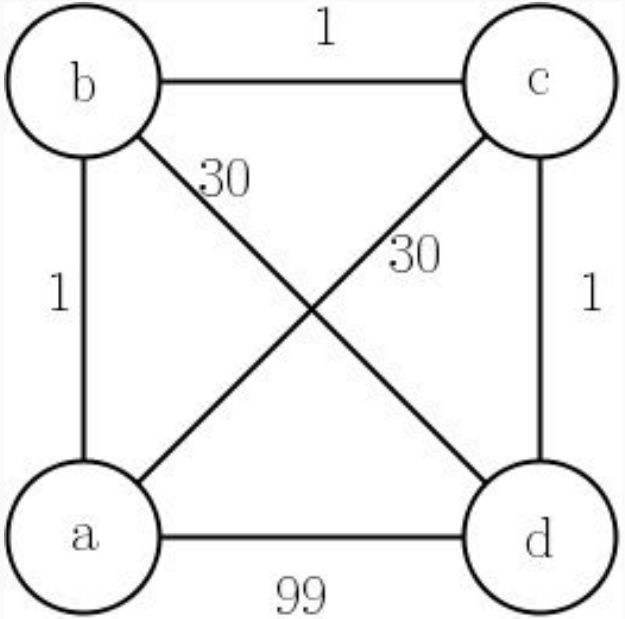
Non-Differentiable Optimization

RL for the Travelling Salesman Problem (TSP)

Definition (TSP):

In TSP, we want to find a Hamiltonian cycle in a complete graph with the shortest length.

I.e. We want to visit all notes in the shortest amount of time.

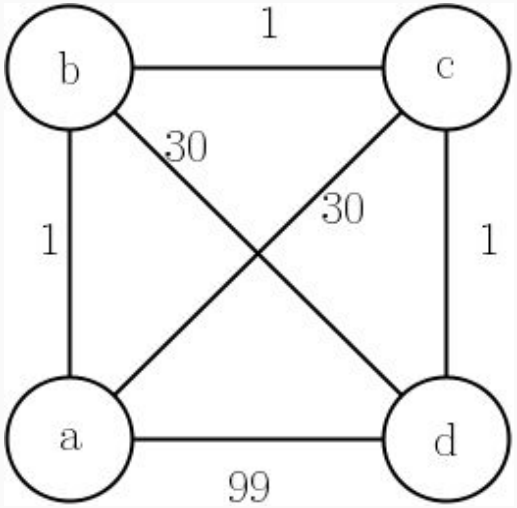


TSP as a RL problem: Attention, Learn To Solve Routing Problems! [Kool et al., 2019]

- State s is a graph with n nodes. (n is a fixed constant.)
- Action a = (a_1, a_2, ..., a_n) is a permutation of the nodes (i.e. a Hamiltonian cycle).
- Policy with weights θ is given by:

$$p_\theta(a|s) = \prod_{t=1}^{n} p_\theta(a_t|s, a_{1:t-1})$$

where $p_\theta(a_t|s, a_{1:t-1})$ is implemented by a modified transformer network.



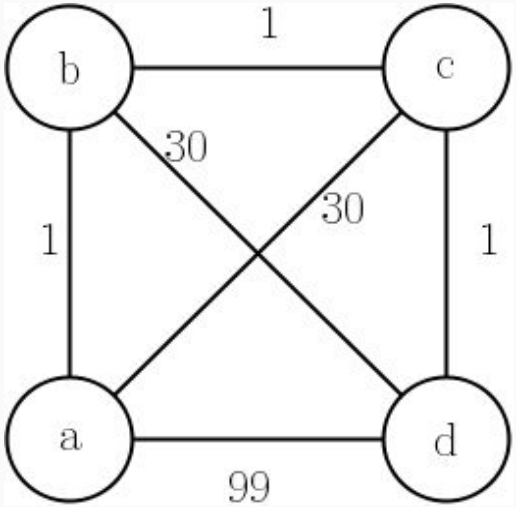
TSP as a RL-Problem: Attention, Learn To Solve Routing Problems! [Kool et al., 2019]

- Loss is defined by the weight of the loop:

  $L(a) = \sum_{i=1}^{n} w_{(a_i, a_{i+1})}$

- Update the parameters θ using REINFORCE:

  $\nabla L(\theta|s) = \mathbb{E}_{p_\theta(a|s)} L(a) \nabla \log p_\theta(a|s)$

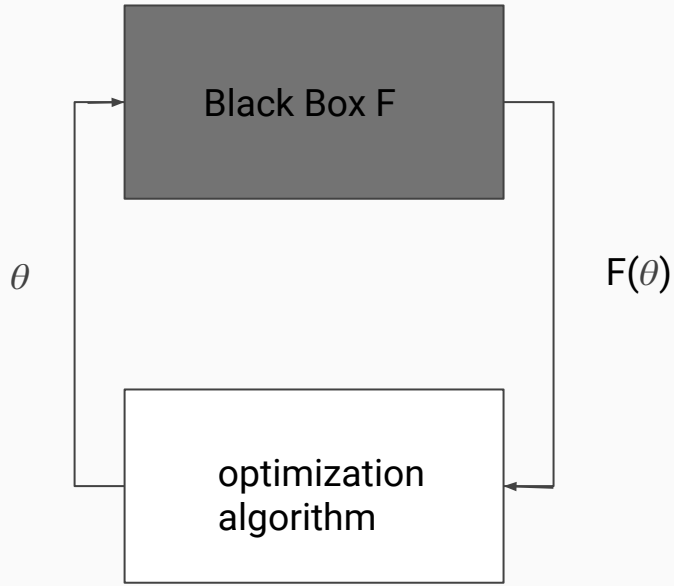


Experiments

	Method	$n = 20$ Obj.	$n = 20$ Gap	$n = 20$ Time	$n = 50$ Obj.	$n = 50$ Gap	$n = 50$ Time	$n = 100$ Obj.	$n = 100$ Gap	$n = 100$ Time
TSP	Concorde	3.84	0.00%	(1m)	5.70	0.00%	(2m)	7.76	0.00%	(3m)
	LKH3	3.84	0.00%	(18s)	5.70	0.00%	(5m)	7.76	0.00%	(21m)
	Gurobi	3.84	0.00%	(7s)	5.70	0.00%	(2m)	7.76	0.00%	(17m)
	Gurobi (1s)	3.84	0.00%	(8s)	5.70	0.00%	(2m)		-	
	Nearest Insertion	4.33	12.91%	(1s)	6.78	19.03%	(2s)	9.46	21.82%	(6s)
	Random Insertion	4.00	4.36%	(0s)	6.13	7.65%	(1s)	8.52	9.69%	(3s)
	Farthest Insertion	3.93	2.36%	(1s)	6.01	5.53%	(2s)	8.35	7.59%	(7s)
	Nearest Neighbor	4.50	17.23%	(0s)	7.00	22.94%	(0s)	9.68	24.73%	(0s)
	Vinyals et al. (gr.)	3.88	1.15%		7.66	34.48%			-	
	Bello et al. (gr.)	3.89	1.42%		5.95	4.46%		8.30	6.90%	
	Dai et al.	3.89	1.42%		5.99	5.16%		8.31	7.03%	
	Nowak et al.	3.93	2.46%			-			-	
	EAN (greedy)	3.86	0.66%	(2m)	5.92	3.98%	(5m)	8.42	8.41%	(8m)
	AM (greedy)	**3.85**	**0.34%**	(0s)	**5.80**	**1.76%**	(2s)	**8.12**	**4.53%**	(6s)

Evolutionary Strategies for Optimizing RL Problems

Black-Box Optimization



Evolutionary Strategies (ES)

initialize ψ
repeat:

- Generate a set of samples D = {(θ_1, F(θ_1)), …, (θ_n, F(θ_n))}, where θ_i is drawn from the distribution $p_\psi(\theta)$
- Evaluate the fitness of samples in D
- Use the fitness to update the parameter ψ

Example: Simple Gaussian ES

Define ψ = (μ, σ) and θ_i ~ N(μ, σ^2I),

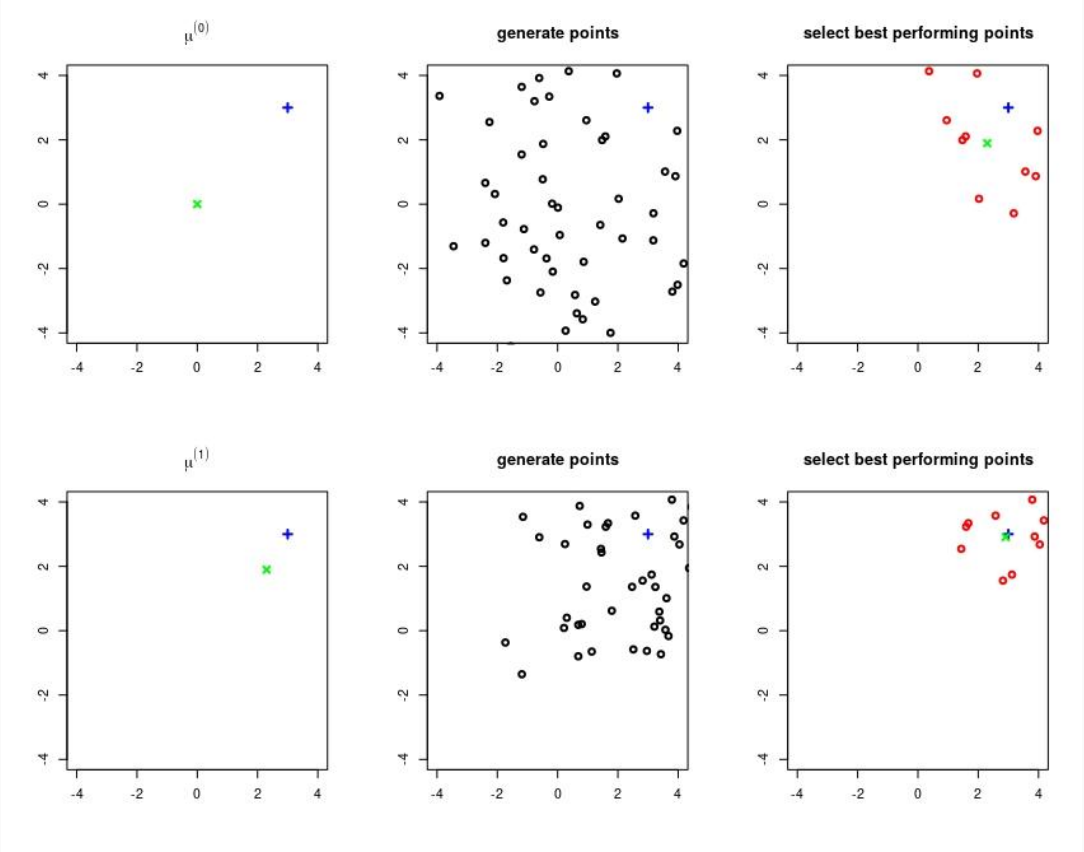
Initialize the parameters $\psi^{(0)}$ = ($\mu^{(0)}$, $\sigma^{(0)}$), size of the elite set m ϵ {1, ..., n}
repeat for t ϵ {1, ..., T}:

- Generate a set of samples D = {(θ_1, F(θ_1)), ..., (θ_n, F(θ_n))}, where θ_i ~ N($\mu^{(0)}$, $\sigma^{(0)2}$I)
- Let L be the set of θ_i with the largest values of F(θ_i) with |L| = m
- Update $\psi^{(t+1)}$ = ($\mu^{(t+1)}$, $\sigma^{(t+1)}$) = (mean(L), std(L))

Example: Simple Gaussian ES

Want to find the minimum of F(θ) = distance of θ to +

x = current mean
o = generated points with mean x and std σ
o = best performing points



Recap REINFORCE

In policy gradient RL, we are optimizing $v_\theta(s) = \mathbb{E}_{\pi_\theta}[G|s] = \mathbb{E}_{\pi_\theta}[\sum_{t=0}^{T} \gamma^t r_t|s]$

We use the log-likelihood trick in REINFORCE to get:

$\nabla_\theta v_\theta(s) \propto \mathbb{E}_{\pi_\theta}[G_t \nabla_\theta \log(\pi_\theta(a, s))]$

Evolutionary Strategies as a Scalable Alternative for RL [Salimans et al., 2017]

In contrast to REINFORCE, which optimizes $v_\theta(s) = \mathbb{E}_{\pi_\theta}[G|s]$, we will optimize the following expectation with respect to the parameter ψ:

$$\mathbb{E}_{\theta \sim p_\psi}[F(\theta)]$$

where $F(\theta) = G_\theta$. Here G_θ is the cumulative reward of one episode following policy π_θ and an unbiased estimate of the value function $v_\theta(s_0) = \mathbb{E}_{\pi_\theta}[G|s_0]$

Evolutionary Strategies as a Scalable Alternative for RL [Salimans et al., 2017]

Say, we want to model θ by a Gaussian with mean ψ and fixed covariance σI. I.e. $\theta \sim p_\psi = N(\psi, \sigma^2 I)$
Then we can write the gradient as follows:

$$
\begin{aligned}
\nabla_\psi \mathbb{E}_{\theta \sim p_\psi}[F(\theta)] &= \mathbb{E}_{\theta \sim p_\psi}[F(\theta) \nabla_\psi \log p_\psi(\theta)] \\
&\propto \mathbb{E}_{\theta \sim p_\psi}[F(\theta) \nabla_\psi (-\frac{1}{2} \frac{(\theta - \psi)^2}{\sigma^2})] \\
&= \mathbb{E}_{\theta \sim p_\psi}[F(\theta) (\frac{(\theta - \psi)}{\sigma^2})] \\
&= \mathbb{E}_{\epsilon \sim N(0,I)}[F(\psi + \epsilon\sigma) (\frac{(\epsilon\sigma)}{\sigma^2})] \\
&= \mathbb{E}_{\epsilon \sim N(0,I)}[F(\psi + \epsilon\sigma) (\frac{\epsilon}{\sigma})] \\
&= \frac{1}{\sigma} \mathbb{E}_{\epsilon \sim N(0,I)}[\epsilon F(\psi + \epsilon\sigma)]
\end{aligned}
$$

Evolutionary Strategies as a Scalable Alternative for RL [Salimans et al., 2017]

Recalling the definition of directional derivatives in higher dimensions,

$$D_v f(x) = \lim_{h \to 0} \frac{f(x+hv) - f(x)}{h}$$

we see that our gradient can be interpreted as randomized finite differences:

$$\frac{1}{\sigma} \mathbb{E}_{\epsilon \sim N(0,I)} [\epsilon F(\psi + \epsilon\sigma)]$$
$$= \mathbb{E}_{\epsilon \sim N(0,I)} [\epsilon \frac{F(\psi + \epsilon\sigma) - F(\psi)}{\sigma}]$$

Evolutionary Strategies as a Scalable Alternative for RL [Salimans et al., 2017]

Algorithm 2 Parallelized Evolution Strategies

Input: Learning rate α, noise standard deviation σ, initial policy parameters ψ_0
Initialize: n workers with known random seeds, and initial parameters ψ_0
for $t = 0, 1, 2, \ldots$ **do**
  for each worker $i = 1, \ldots, n$ **do**
    Sample $\epsilon_i \sim \mathcal{N}(0, I)$
    Compute returns $F_i = F(\psi_t + \sigma\epsilon_i)$
  end for
  Send all scalar returns F_i from each worker to every other worker
  for each worker $i = 1, \ldots, n$ **do**
    Reconstruct all perturbations ϵ_j for $j = 1, \ldots, n$ using known random seeds
    Set $\psi_{t+1} \leftarrow \psi_t + \alpha \frac{1}{n\sigma} \sum_{j=1}^{n} F_j \epsilon_j$
  end for
end for

Advantages of ES

- Efficient parallelization
  - by synchronizing random seeds of workers before training, each worker knows what perturbation the others used
  - then we only need to communicate the episode return between workers
- computation and memory efficient since no backpropagation is needed
- robust: little hyperparameter tuning needed, no frameskip needed
- easy to implement

But

- 3-10 times less data efficient

Evolutionary Strategies as a Scalable Alternative for RL [Saliman et al., 2017]

Experiments on MujoCo:

Ratio of ES timesteps to TRPO timesteps needed to reach various percentages of TRPO's learning progress at 5 million timesteps.

Environment	25%	50%	75%	100%
HalfCheetah	0.15	0.49	0.42	0.58
Hopper	0.53	3.64	6.05	6.94
InvertedDoublePendulum	0.46	0.48	0.49	1.23
InvertedPendulum	0.28	0.52	0.78	0.88
Swimmer	0.56	0.47	0.53	0.30
Walker2d	0.41	5.69	8.02	7.88

Possible Explanation

While exploration in RL is done in the action space, exploration in ES is done in the parameter space.

I.e. In ES, we explore by changing the whole parameter θ (not just by making actions more random), potentially leading to a new behaviour.

Genetic Algorithms for Optimizing RL Problems

Definition (Genetic Algorithms)

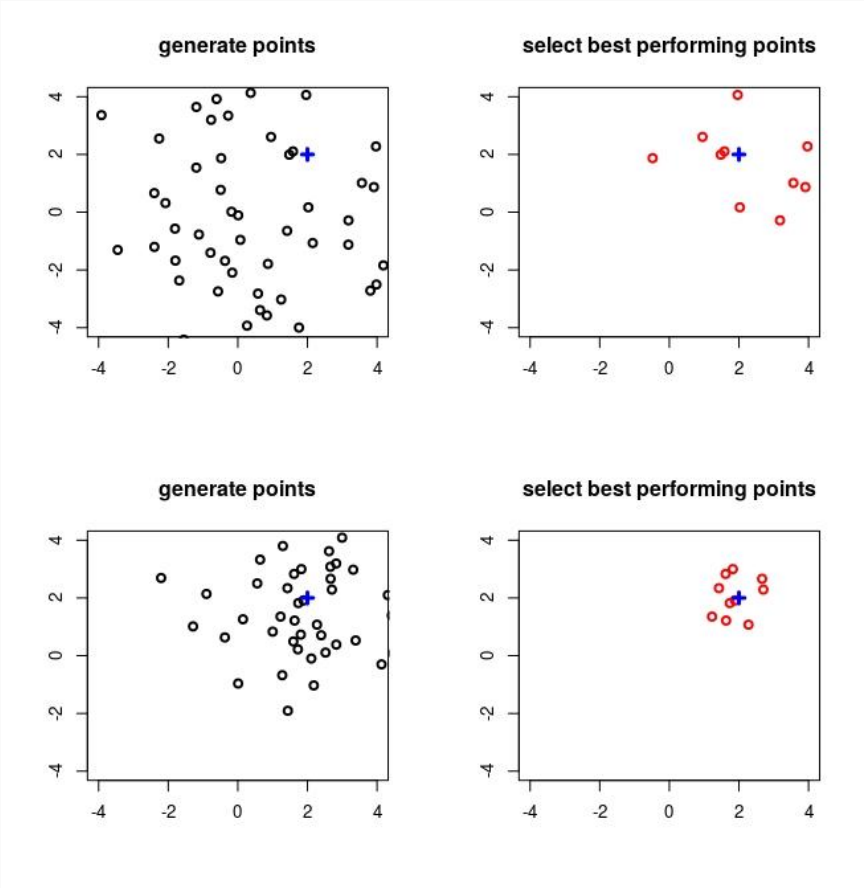
initialize parameter vectors $\theta_1^{(0)}$, ..., $\theta_n^{(0)}$
repeat for t in {1, ..., T}:

- Generate a set of samples D = {($\theta_1^{(t)}$, F($\theta_1^{(t)}$)), ..., ($\theta_n^{(t)}$, F($\theta_n^{(t)}$))}
- Evaluate the fitness of samples in D
- Select the B top performing parameter vectors; they become the parents
- Mutate and/or breed the parents in some way to get new parameters $\theta_1^{(t+1)}$, ..., $\theta_n^{(t+1)}$

Example GA

We want to find the minimum of F(θ) = distance of θ to +

o = child points
o = best performing points / parents of the next generation



Simples Gaussian GA for RL Problems [Petroski Such et al., 2017]

initialize parameter vectors $\theta_1^{(0)}$, …, $\theta_n^{(0)}$, the size of the elite set m, standard deviation σ

repeat for t in {1, ..., T}:

- For each $\theta_i^{(t)}$ do 30 runs in the environment with the policy $\pi(\theta_i^{(t)})$ and store the mean of the cumulative rewards G_n.
- Let L be the set of the m parameters with the highest mean of rewards.
- for i in {1, …, n}:
  - choose θ in L uniformly at random
  - Define $\theta_i^{(t+1)} = \theta + \sigma\epsilon$, where $\epsilon \sim N(0, 1)$

Advantages of ES and GA

Same advantages as ES:

- Efficient parallelization
- computation and memory efficient since no backpropagation is needed
- robust: little hyperparameter tuning needed, no frameskip needed
- easy to implement

Experiments

GA, ES run significantly faster than DQN and A3C.
GA outperformed the other methods on frostbite, skiing and venture.

We see: Each method outperforms all other methods on at least one game.

When RS performs well, so does GA.

	DQN	ES	A3C	RS	GA
Frames	200M	1B	1B	1B	1B
Time	~7-10d	~ 1h	~ 4d	~ 1h or 4h	~ 1h or 4h
Forward Passes	450M	250M	250M	250M	250M
Backward Passes	400M	0	250M	0	0
Operations	1.25B U	250M U	1B U	250M U	250M U
amidar	**978**	112	264	143	263
assault	4,280	1,674	**5,475**	649	714
asterix	4,359	1,440	**22,140**	1,197	1,850
asteroids	1,365	1,562	**4,475**	1,307	1,661
atlantis	279,987	**1,267,410**	911,091	26,371	76,273
enduro	**729**	95	-82	36	60
frostbite	797	370	191	1,164	**4,536**
gravitar	473	**805**	304	431	476
kangaroo	7,259	**11,200**	94	1,099	3,790
seaquest	**5,861**	1,390	2,355	503	798
skiing	-13,062	-15,443	-10,911	-7,679	†**-6,502**
venture	163	760	23	488	**969**
zaxxon	5,363	6,380	**24,622**	2,538	6,180

Possible Explanation

It could be that saddle points or noisy gradients prevent the gradient-based methods from learning effectively in some environments.

Since GA and RS do not use gradients, they are not affected by this.

Conclusion

- Some optimization problems can be framed as RL-tasks and then solved with RL methods
- Black Box optimization methods like GA and ES can provide an alternative approach to solve RL problems

References

- Wouter Kool, Herke van Hoof, and Max Welling. *Attention, learn to solve routing problems! International Conference on Learning Representations,* 2019. URL: https://openreview.net/forum?id=ByxBFsRqYm.
- Salimans, T., Ho, J., Chen, X., and Sutskever, I. *Evolution Strategies as a Scalable Alternative to Reinforcement Learning.* ArXiv e-prints, 2017. URL: https://arxiv.org/abs/1703.03864; Blog-Post: https://openai.com/blog/evolution-strategies/
- Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, Jeff Clune. *Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning*. arXiv e-prints, 2017. URL: https://arxiv.org/abs/1712.06567