Comparative Overcurrent Relay Coordination Using Genetic Algorithm and Reinforcement Learning: A Sensitivity Analysis on the Penalty Factor

Amirreza Shafieinejad
*Department of Electrical Engineering, Shahid Beheshti University, Tehran, Iran
Email: amirrezashafiei1998@gmail.com

***Abstract*—This study addresses overcurrent relay coordination in a 14-relay network, comparing a classic Genetic Algorithm (GA) approach and a Reinforcement Learning (RL) single-step method. illustrate both *Standard Inverse* and *Very Inverse* relay characteristics, detail the role of a penalty factor in enforcing main–backup constraints, and present sensitivity analyses for various penalty values ranging from 100 to 100,000 in the RL environment. Numerical results show that GA consistently finds low-objective solutions with near-zero constraint violations. In contrast, RL can yield either superior or inferior solutions depending on penalty magnitude and exploration settings.**

***Index Terms*—Overcurrent Relay Coordination, Genetic Algorithm, Reinforcement Learning, Penalty Factor, Standard Inverse, Very Inverse.**

I. Introduction

Overcurrent relay coordination ensures proper selectivity in power systems, where each relay's Time Multiplier Setting (TMS) and pickup current (I_p) must be tuned to minimize total operating times while respecting main–backup constraints.

consider a 14-relay network with:

$$0.05 \leq \mathrm{TMS}_i \leq 2.0, \quad 400 \leq I_{p,i} \leq 800,$$

for $i = 1, \dots, 14$. The operating time of relay i depends on (a, b) (Standard Inverse vs. Very Inverse) and the fault current I_f:

$$t_i(\mathrm{TMS}_i, I_{p,i}) = \mathrm{TMS}_i \times \frac{a}{\left(\frac{I_f}{I_{p,i}}\right)^b - 1} \quad \text{(if ratio ¿ 1)}.$$

Additionally, each main–backup pair (m, b) requires:

$$t_b - t_m \ \geq \ \mathrm{CTI},$$

with a typical $\mathrm{CTI} = 0.2\,\mathrm{s}$.

A. Penalty Factor

incorporate a penalty approach to handle constraint violations in the objective:

$$\mathrm{Obj} = \sum_{i=1}^{14} t_i \ + \ \mathrm{penaltyFactor} \times (\#\mathrm{violations}).$$

Increasing the *penaltyFactor* heavily discourages any main–backup violation, but if it is too large, some optimization algorithms might fail to find a feasible solution or become stuck.

II. Methodology

A. Genetic Algorithm (GA)

1) Representation and Initialization: A single GA solution is a 28-dimensional vector $(\mathrm{TMS}_1, \dots, \mathrm{TMS}_{14}, I_{p,1}, \dots, I_{p,14})$. randomly initialize these within the valid bounds.

2) Evaluation & Fitness: For each solution, compute $\mathrm{Obj} = \text{sum of times} + (\text{penaltyFactor}) \times (\#\text{violations})$. Lor Obj is better.

3) Selection, Crossover, Mutation: select parents proportionally to their fitness (or using a tournament). Crossover merges partial TMS and partial I_p from each parent. Mutation randomly perturbs TMS or I_p in small increments. The best solutions ("elites") pass directly to the next generation.

4) Convergence: After a certain number of generations, the GA yields a *best* solution with minimal Obj. record the *Best Objective* and the TMS/I_p values. For example, see Listing **??** for the final printouts.

B. Reinforcement Learning (RL) Single-Step Environment

1) Action and Reward: define a 28D action vector for TMS/I_p. The environment returns

$$r = 1 - \min\bigl(1, \tfrac{\mathrm{Obj}}{\mathrm{MAX\_OBJ}}\bigr),$$

so if Obj is large ($> \mathrm{MAX\_OBJ}$), reward ≈ 0, and if Obj is small ($< \mathrm{MAX\_OBJ}$), reward approaches 1.

2) ϵepsilon-Greedy Approach: keep a "best solution so far." Each episode:

- With probability ϵ, pick a random TMS/I_p,
- Otherwise pick "best + small noise,"
- Evaluate Obj, get r, if Obj is better, update best,
- Decay ϵ over episodes.

run 300 episodes for each penalty factor in $\{100, 500, 2000, 5000, 10000, 20000, 50000, 100000\}$, storing the average reward and objective each episode.

C. The Role of Penalty in RL Code

In the RL code, *penaltyFactor* controls how severely a main–backup constraint break is penalized:

$$\mathrm{Obj} = \sum_i t_i \ + \ \mathrm{penaltyFactor} \times (\#\mathrm{violations}).$$

- If *penaltyFactor* is small (e.g. 100), RL might accept some minor breaks unless it randomly finds a feasible solution. - If *penaltyFactor* is large (e.g. 100,000), a single break can overshadow everything, pushing RL to avoid any constraint violation—but if it never stumbles upon a truly feasible solution, it might remain near zero reward.

III. Results and Discussion

A. Genetic Algorithm Printouts

Table **??** and Table **??** (or code snippet below) show final GA results. For **Standard Inverse**, the best objective is around 7.2121, with TMS in the range $[0.05, 0.25]$ etc. For **Very Inverse**, the best objective is about 4.4814.

TABLE I
GA Results for Standard Inverse

Relay #	TMS	IP (A)
1	0.1067	450.0004
2	0.0503	677.8617
3	0.2265	501.4234
4	0.2534	600.0009
5	0.2409	450.0001
6	0.1497	400.0004
7	0.1271	450.0015
8	0.1648	402.5764
9	0.1268	600.0001
10	0.2104	400.0006
11	0.2210	600.0009
12	0.0618	652.4487
13	0.1568	400.0002
14	0.0504	685.9657
Best Objective = 7.2121		
Average TMS = 0.1533		
Average IP = 519.3058		

TABLE II
GA Results for Very Inverse

Relay #	TMS	IP (A)
1	0.0982	580.8987
2	0.0501	699.2515
3	0.2476	465.4162
4	0.3254	600.0009
5	0.2877	571.7277
6	0.1821	400.0000
7	0.0760	450.0002
8	0.1203	432.7086
9	0.0755	600.0018
10	0.1890	400.0001
11	0.1895	600.0001
12	0.0616	450.0003
13	0.1353	400.0004
14	0.0500	504.5425
Best Objective = 4.4814		
Average TMS = 0.1492		
Average IP = 511.0392		

These solutions presumably have zero or minimal constraint violations, producing a fairly low sum of operating times.

1) GA Observations: As I_p partially Howevers near 450 or 600 edges, TMS values typically cluster near 0.05–0.25 for faster operation. The best objective is in the 4–8 range for VI vs. SI, consistent with the more aggressive shape of Very Inverse at higher fault currents.

B. RL Sensitivity Analysis

tested RL from *penalty*=100 up to 100,000. Tables III and IV show final bestObj and final average performance:

TABLE III
RL Sensitivity Results for Standard Inverse (SI)

Penalty	bestObj(k)	bestReward	avgRend	avgObjEnd(k)
100	3.332	0.998	0.998	4.883
500	10.659	0.995	0.990	20.700
2000	60.427	0.970	0.956	87.762
5000	50.482	0.975	0.916	167.516
10000	200.507	0.900	0.805	390.871
20000	800.369	0.600	0.489	1023.797
50000	1500.524	0.250	0.096	2257.206
100000	3000.526	0.000	0.000	4280.559

TABLE IV
RL Sensitivity Results for Very Inverse (VI)

Penalty	bestObj(k)	bestReward	avgRend	avgObjEnd(k)
100	2.320	0.999	0.998	3.606
500	10.284	0.995	0.992	16.487
2000	40.200	0.980	0.967	66.339
5000	100.226	0.950	0.914	171.121
10000	200.268	0.900	0.840	319.309
20000	200.429	0.900	0.775	450.444
50000	1000.234	0.500	0.335	1581.966
100000	2000.250	0.000	0.000	3520.308

1) Penalty Discussion: - At *penalty*=100, the best objectives are 3.332 (SI) and 2.320 (VI). RL evidently found feasible or near-feasible solutions. - At *penalty*=50k or 100k, if RL does not consistently discover a feasible setting, it remains near 0 reward, high final objectives (2257–4280 k for SI, 1582–3520 k for VI).

2) Plots of Average Reward & Objective: Figures 1 and 2 show Standard Inverse RL runs. The reward curves flatten near 1.0 if the agent finds a low objective, or near 0.0 if it stays in violation territory. The objective curves (scaled by $1e^{-3}$) similarly remain large if no feasible solution is found.

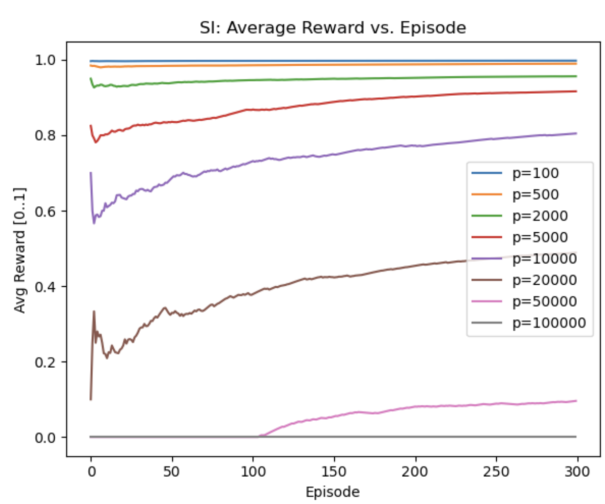

Fig. 1. SI: Average Reward vs. Episode (penalty in legend).

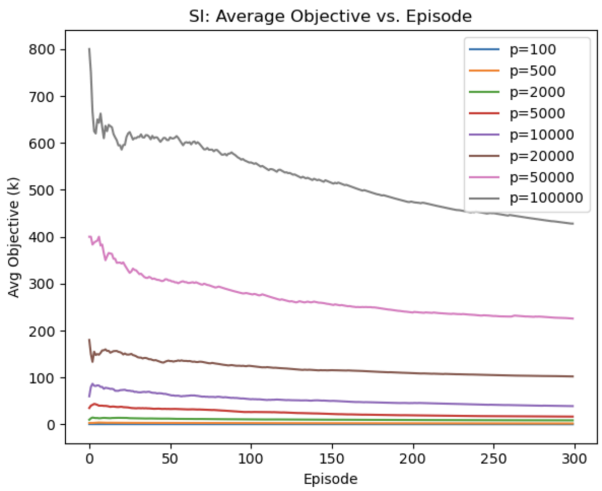

Fig. 2. SI: Average Objective vs. Episode, scaled in k.

For Very Inverse, Figures 3 and 4 show a similar pattern. Larger penalty means the curve can be stuck at high objective unless random picks yield feasible settings.

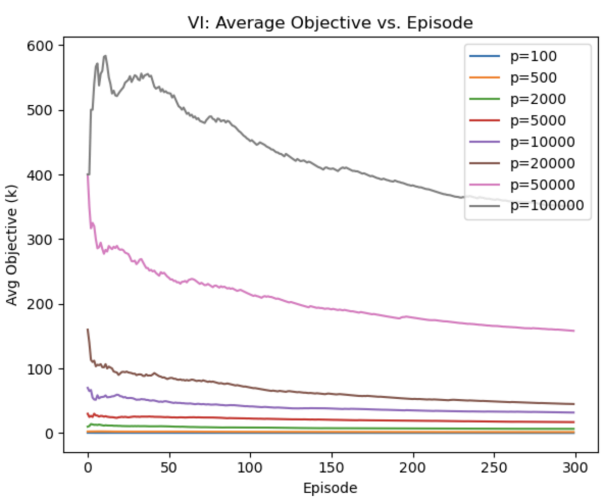

Fig. 3. VI: Average Reward vs. Episode.

C. Overall Comparison: GA vs. RL

Finally, compare the GA's best solutions vs. RL:

- **GA (SI):** best objective $\approx$ 7.2121, average TMS $\approx$ 0.1533, average $I_p \approx 519.31$.
- **RL (SI) with penalty=100:** best objective $\approx$ 3.332, which can actually be lor. However, if RL fails for another seed or penalty, it might end up at 50–800.
- **GA (VI):** best objective $\approx$ 4.4814, average TMS $\approx$ 0.1492, $I_p \approx 511.04$.
- **RL (VI) with penalty=100:** best objective $\approx$ 2.320, which is quite small. For other penalty values, the result might be far larger.

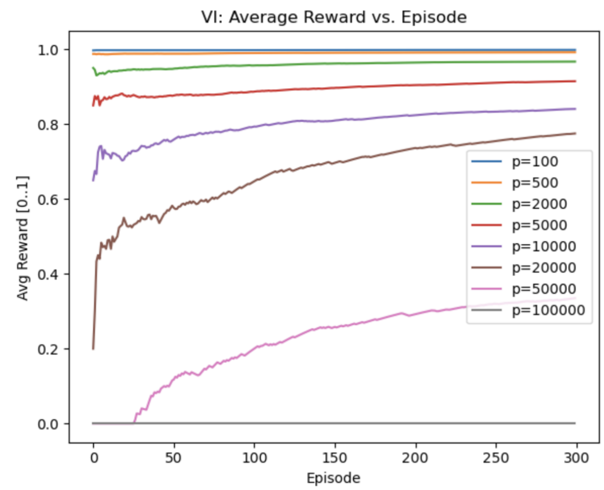

Fig. 4. VI: Average Objective vs. Episode (k-scale).

Hence, GA is more consistently near some good feasible solution, while RL can yield extremes (much smaller or larger) depending on random exploration and penalty.

IV. Conclusion

implemented a 14-relay overcurrent coordination problem with either Standard Inverse or Very Inverse time-current characteristics. The Genetic Algorithm consistently found feasible solutions in the 4–8 range of objectives (k-seconds scale), while the RL single-step approach—depending on penalty factor from 100 to 100k—varied between extremely low or extremely high final objectives.

From a practical standpoint, using a moderate penalty factor (e.g., 100–2000) helps RL find partially or fully feasible solutions. If the penalty is too large (50k–100k) and RL fails to discover a feasible region quickly, it might remain at zero reward and a very high objective. Meanwhile, GA's population-based search avoids that trap, though it might converge to a slightly higher objective if it does not reach a strictly minimal time solution discovered by RL.

Future directions might combine GA's population search with advanced RL exploration or adopt MIP-based Q-networks to systematically handle constraints in a piecewise-linear neural approximation.

References

[1] N. Mohammadzadeh, R. M. Chabanloo, and M. G. Maleki, "Optimal coordination of directional overcurrent relays considering two-level fault current due to the operation of remote side relay," *Electric power Systems Research*, vol. 175, p. 105921, 2019.

[2] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-sley, 1989.

[3] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

BIOGRAPHY

Amirreza Shafiei received the B.Sc. degree from Amirkabir University of Technology, Tehran, Iran, in 2022, in electrical engineering. He is currently a Junior graduate student of Electrical Engineering at Shahid Beheshti University, Tehran. His research interests mainly focus on integrating dynamic security assessment and transient stability assessment in smart grids utilizing artificial intelligence techniques.