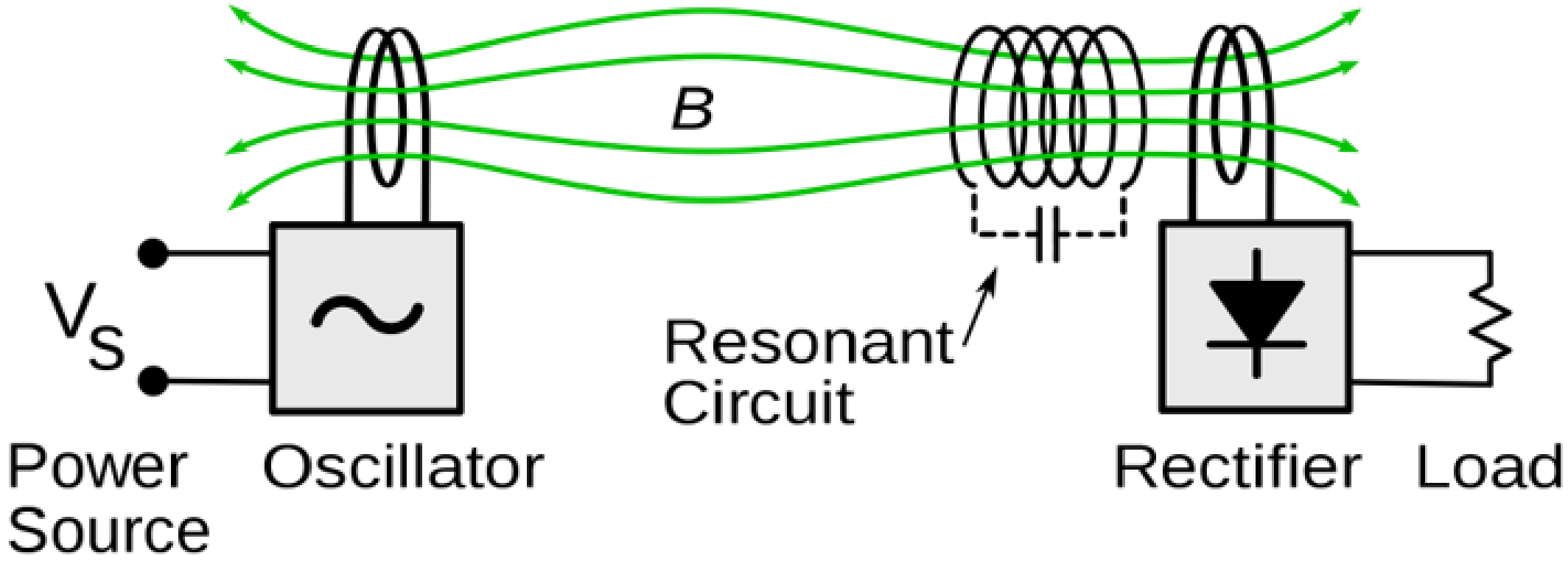
UNIVERSIDADE FEDERAL DO ABC
NF317E - Tópicos em Inteligência Artificial: Aprendizagem Profunda

Licia Sales Costa Lima

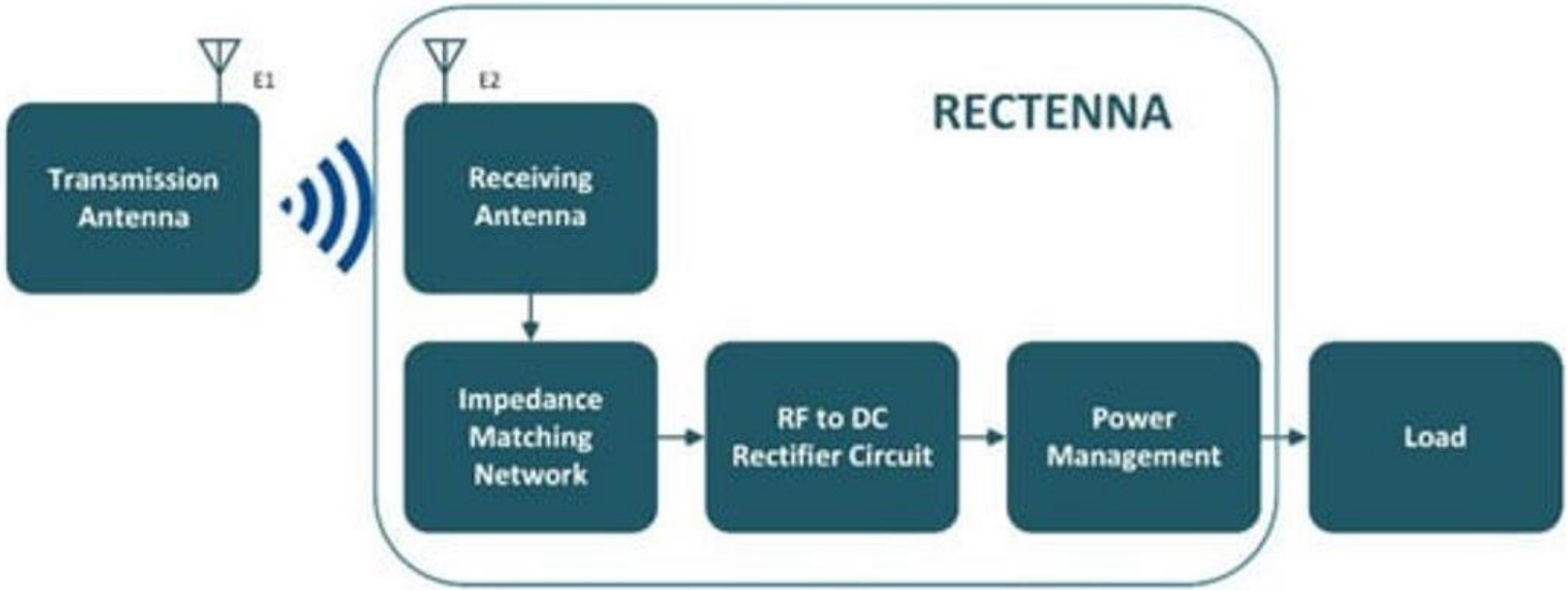
OTIMIZAÇÃO DE MODELO DE CIRCUITO RETIFICADOR DE RECTENNA USANDO APRENDIZAGEM POR REFORÇO

O que é Wireless Power Transfer



UFABC

Circuito Rectenna

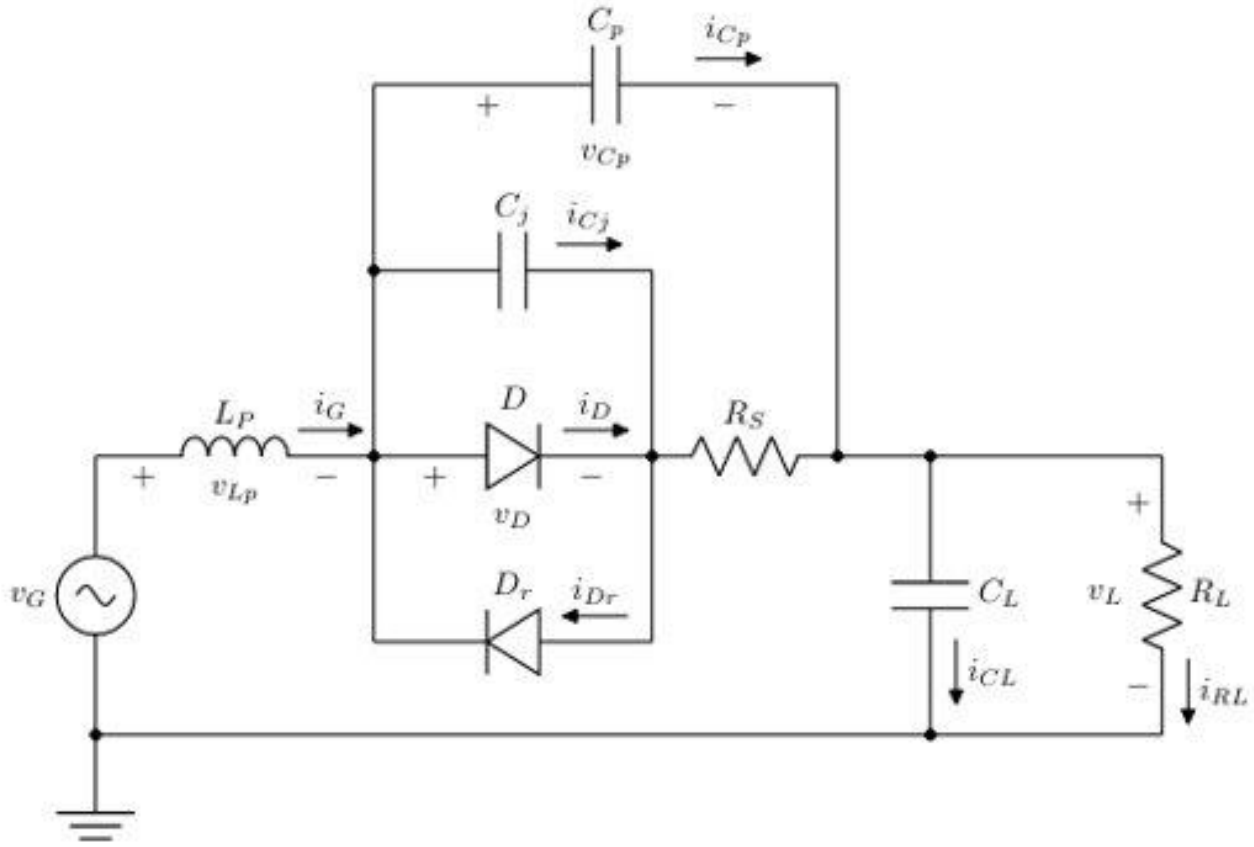


Caracterização do Diodo

Parâmetro	Valor mínimo	Valor Máximo
Capacitância Parasita (CP)	0.05e-12	0.25e-12
Indutância Parasita (LP)	0.8e-9	20e-9
Corrente de Saturação (Is)	1e-8	1e-4
Resistência Parasita (Rs)	5	30
Capacitância de polarização de junção zero (CJ0)	0.1e-12	0.2e-12
Coeficiente de classificação da junção (M)	0.3	0.5
Potencial de junção (Vj)	0.3	0.65
Energia de ativação (EG)	0.69	0.69
Corrente de ruptura reversa (IBV)	0.1e-3	0.1e-3
Capacitância da carga (CL)	1e-12	100e-12
Resistência da carga (RL)	100	50000

UFABC

Circuito equivalente para simulação



UFABC

Curva característica do Diodo

INFLUÊNCIA DA TEMPERATURA

A temperatura é um dos fatores que mais influenciam no funcionamento de um diodo; com o aumento da temperatura a tensão direta (V_D) diminui e a corrente reversa (I_R) aumenta. Isto pode ser observado na figura abaixo.

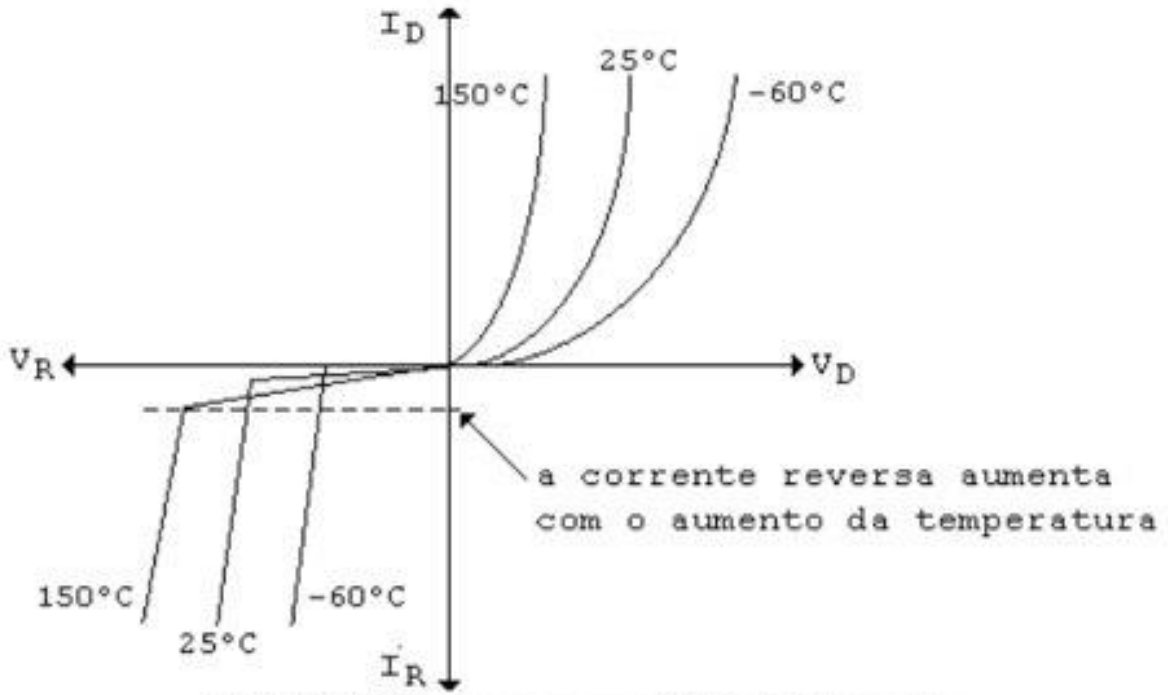
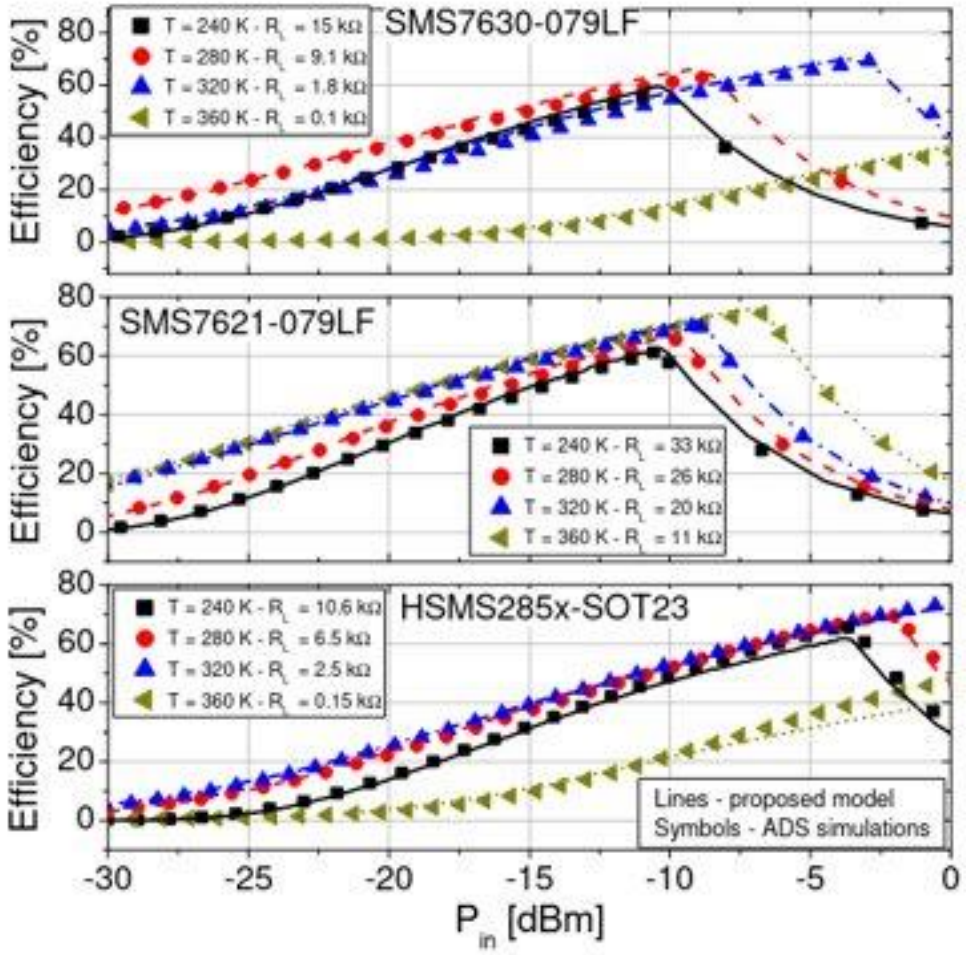
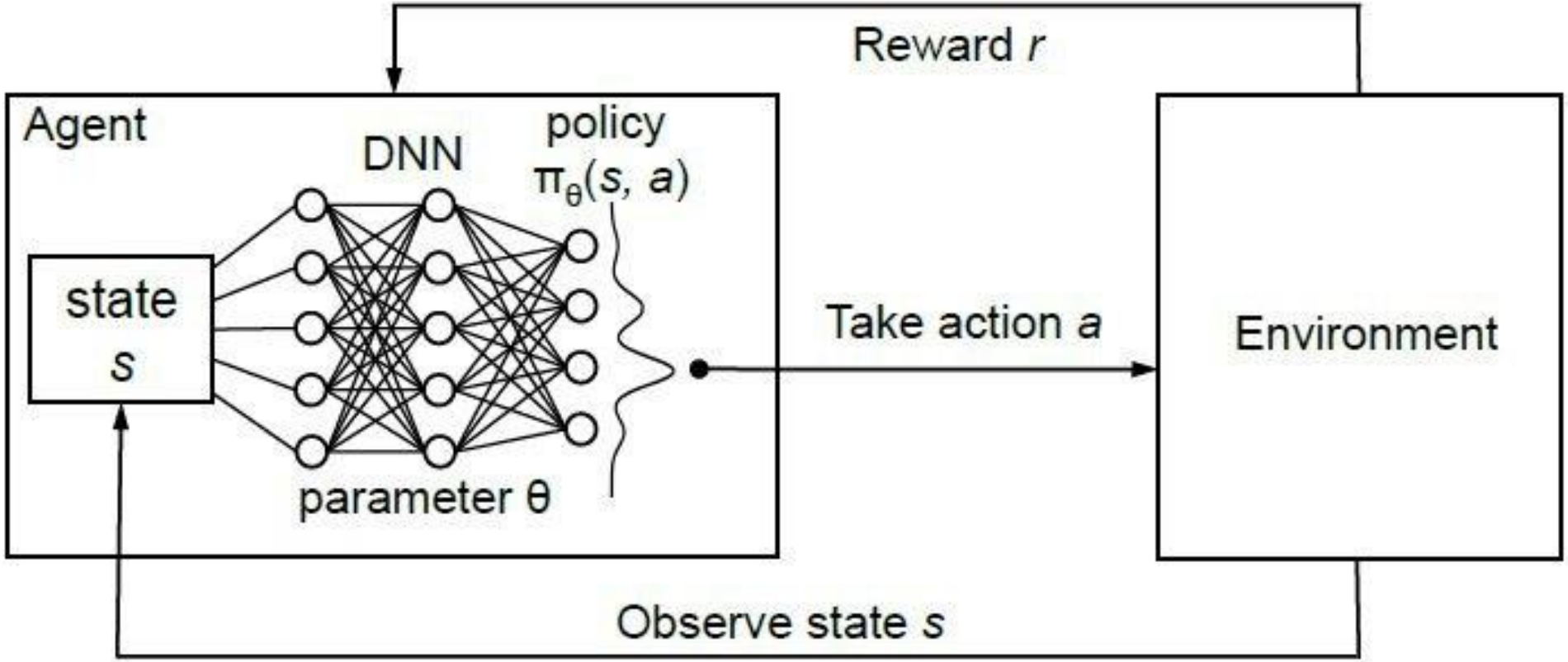


FIGURA 1.13 Variação nas curvas por influência da temperatura

Eficiência em relação a temperatura

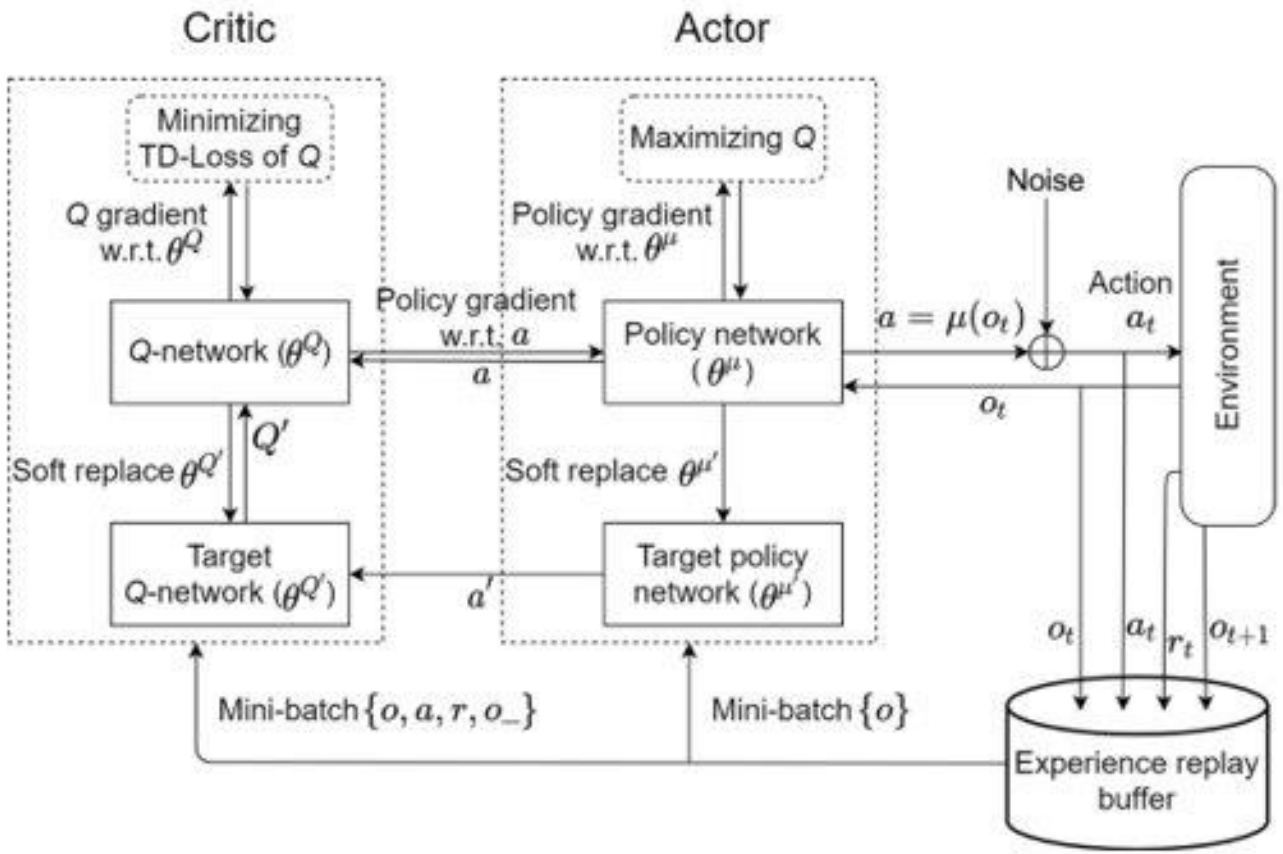


Deep Reinforcement Learning



UFABC

Advantage Actor Critic agent (A2C)



Pseudo-código usado como base

Algorithm 1: DRiLLS Framework

Input : Design, Primitive Transformations
Output: Optimization_Flow

```
env = Initialize(LS_Env);
agent = Initialize(A2C);
for episode = 1 to N do
    episode_design_states = [];
    optimization_sequence = [];
    synth_rewards = [];
    design_state = env.reset();
    for iteration = 1 to k do
        opt_probs = agent.ActorForward(design_state);
        primitive_opt = RandomChoice(opt_prob);
        [next_design_state, synth_reward] =
        env.perform(primitive_opt);
        episode_design_states.append(design_state);
        optimization_sequence.append(primitive_opt);
        synth_rewards.append(synth_reward);
        design_state = next_design_state;
    end
    episode_rewards = DiscountRewards(synth_rewards,
     gamma)
    loss = agent.OptimizerForward(episode_design_states,
     optimization_sequence, episode_rewards);
    agent.update(loss);
    log(episode);
end
```

https://github.com/scale-lab/DRiLLS

https://ieeexplore.ieee.org/abstract/document/9045559

UFABC

Afinal, como foi implementado?

1 – Representação dos Estados:

Os estados representam a reação do ambiente diante de uma otimização sugerida

O Vetor de estados é uma representação do projeto do circuito a uma determinada etapa de otimização.

2 - Espaço de otimização:

O agente explora o espaço de busca = {Parametro_atual, Parametro_atual-z, Eficiencia_atual, Eficiencia_atual -z}.

Essas transformações manipulam o vetor de estado e são usados na função recompensa

3 - Função de recompensa:

Definimos uma função de recompensa lenvando em consideração a Eficiencia media do circuito.

Caso o valor da Eficiencia media aumente, o agente e recompensado, caso a eficiencia diminua, o agente e penalizado, caso a eficiencia se mantenha estagnada, o agente nao e penalizado nem recompensado.

4 - Treinamento do agente A2C:

O agente híbrido tem duas redes, uma baseada em valor e outra baseada em politica, chamada de rede ator e crítico.

Ambas as redes têm uma camada de entrada de tamanho igual ao comprimento do vetor de estados.

A recompensa r é passada para a rede crítica para treinamento e um desconto a recompensa é passada para a rede do ator.

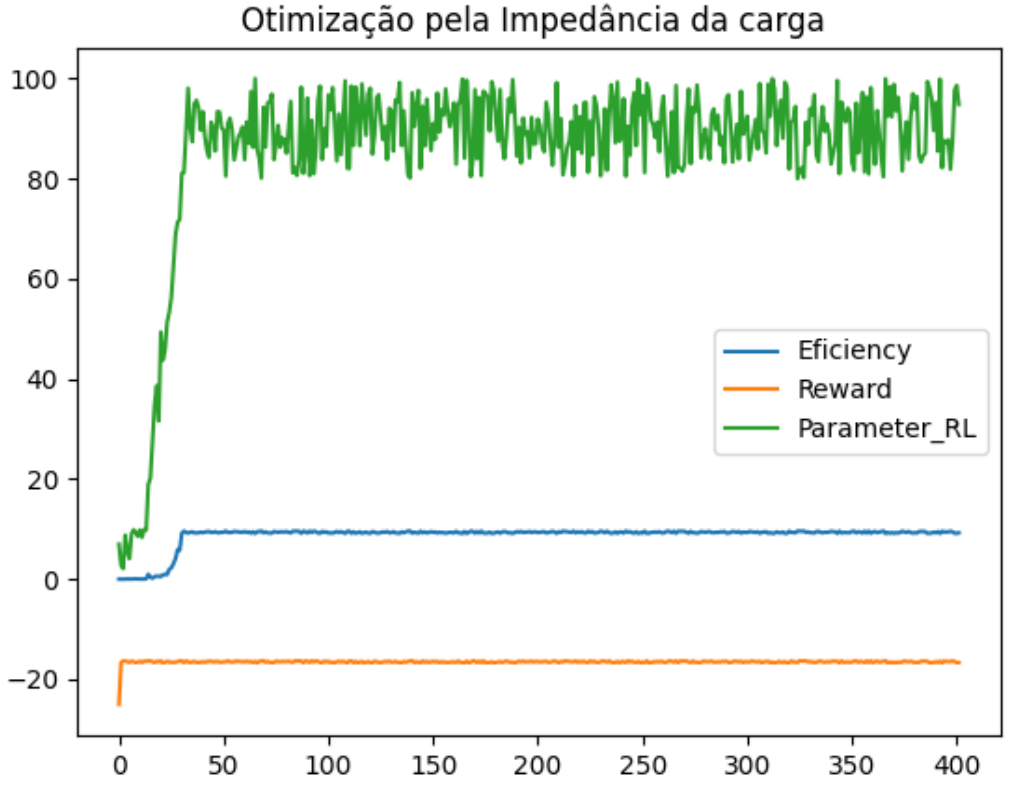
A rede do ator produz a distribuição de probabilidade sobre as transformações disponíveis.

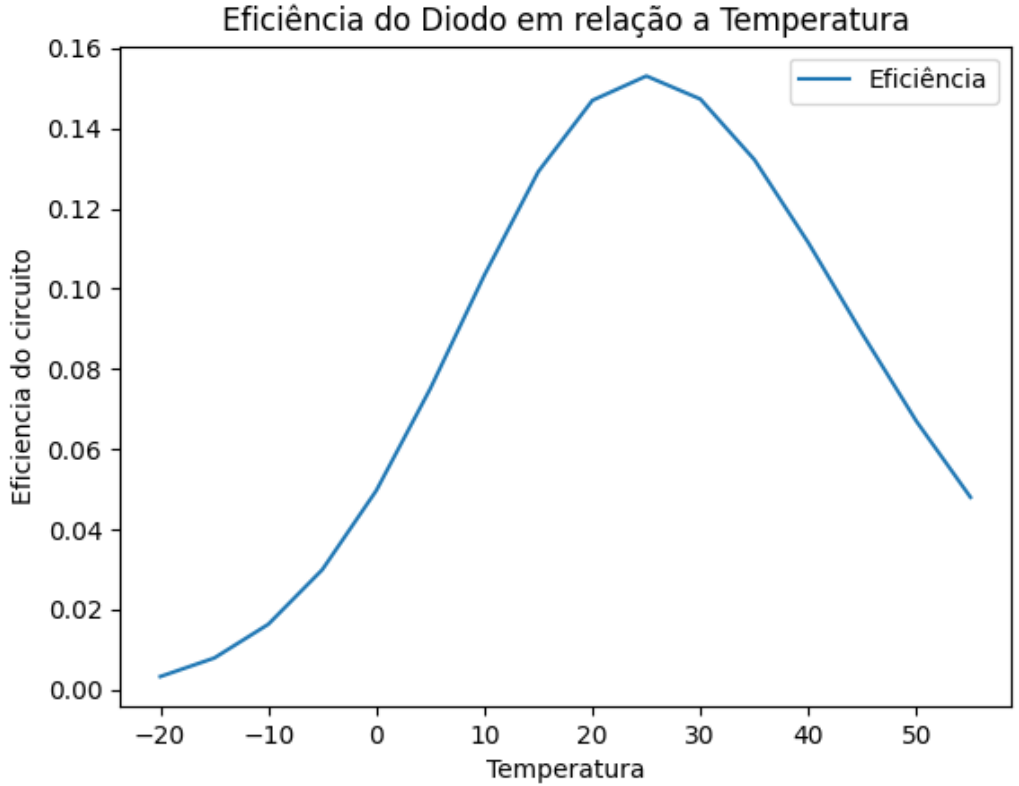
Parâmetros de ambas as redes são atualizados para reduzir a funcao loss usando um otimizador baseado em gradiente.

A escolha de uma arquitetura híbrida de aprendizagem é adequada para tarefas de otimização combinatória pois dá ao agente a oportunidade de explorar diversas combinações de otimizações

UFABC

Resultados obtidos

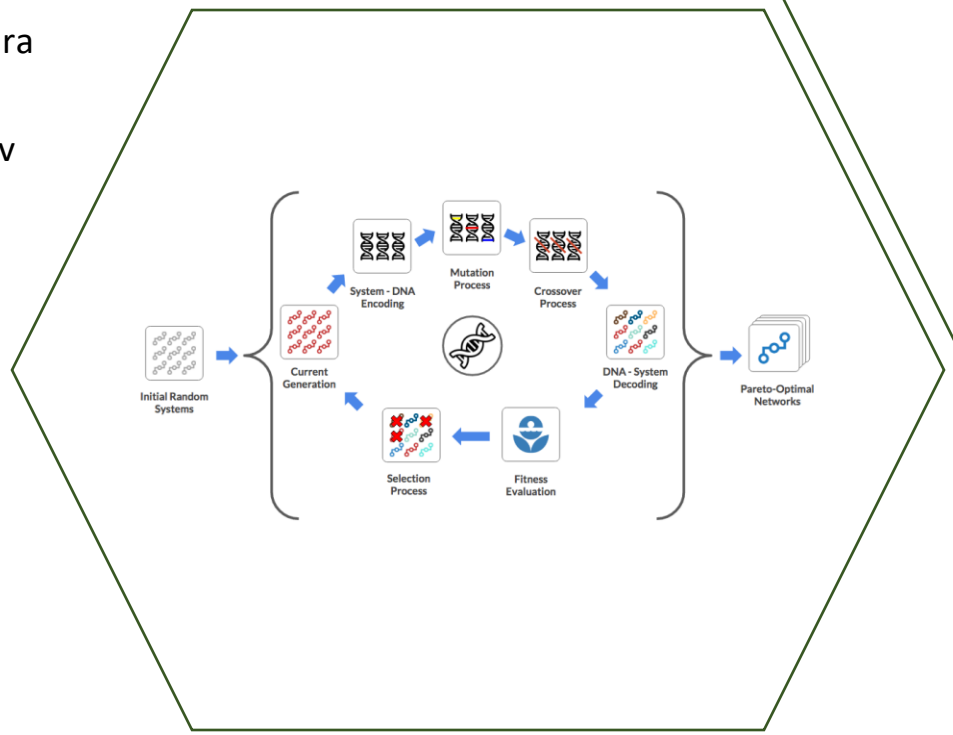


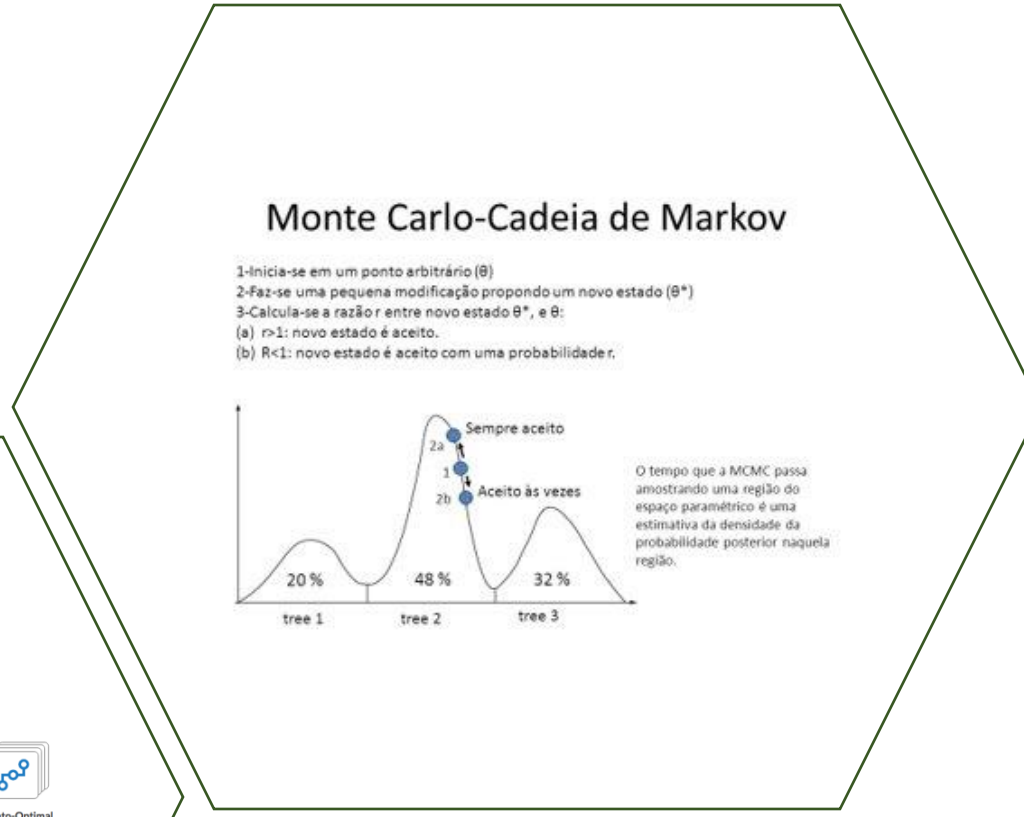


UFABC

Próximos passos

- - Otimizar o ambiente de simulação para executar em paralelo, se possivel, na GPU
- - Atualizar o algoritmo para ter compatibilidade com o TensorFlow 2
- - Incluir os outros parametros de caracterização do diodo na rede
- - Implementar outras estrategias para escolha dos parametros:
- Monte Carlo – Cadeia de Markov
- Algoritmo genetico





Obrigada!

Alguma pergunta?

Referências

Trevisoli, Renan, et al. “Modeling Schottky Diode Rectifiers Considering the Reverse Conduction for RF Wireless Power Transfer.” IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 69, no. 3, 1 Mar. 2022, pp. 1732–1736, ieeexplore.ieee.org/document/9507537, 10.1109/TCSII.2021.3102576. Accessed 30 Nov. 2022.

https://ieeexplore.ieee.org/document/9507537

Hosny, Abdelrahman, et al. DRiLLS: Deep Reinforcement Learning for Logic Synthesis.

https://ieeexplore.ieee.org/abstract/document/9045559

GitHub DRiLLS https://github.com/scale-lab/DRiLLS

Meu GitHub https://github.com/liciascl/ML-Retifier/tree/master/aprendizagem_reforco

UFABC