UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA QUÍMICA

TRABALHO DE CONCLUSÃO DE CURSO EM ENGENHARIA QUÍMICA

Otimização de Produção de Lipídeos de Microalgas: Comparativo Entre Métodos Clássicos e Reinforcement Learning

Autor: Felipe Cao

Orientador: Marcelo Farenzena

Porto Alegre, outubro de 2022

Autor: Felipe Cao

Otimização de Produção de Lipídeos de Microalgas: Comparativo Entre Métodos Clássicos e Reinforcement Learning

Trabalho de Conclusão de Curso apresentado à COMGRAD/ENQ da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de Bacharel em Engenharia Química

Orientador: Marcelo Farenzena

Banca Examinadora:

Prof. Dr. Ing., Jorge Otávio Trierweiler, UFRGS

Leonardo Mandler de Marco, M.Sc. , UFRGS

Porto Alegre 2022

AGRADECIMENTOS

Primeiramente agradeço a minha família, principalmente ao meu pai Marne Luciano Cao, e minha mãe Cleusa Rosmari Cao por todo apoio e carinho durante toda a minha jornada de crescimento. Agradeço a UFRGS e a todos os professores que participaram da minha formação, em especial agradeço ao meu orientador Marcelo Farenzena, pela ajuda durante o processo de realização deste trabalho. Por fim, agradeço a todos que de alguma forma participaram deste período tão importante.

RESUMO

Diante do crescimento dos preços do petróleo e da crise ambiental enfrentada nos dias atuais, a busca por outras fontes de combustíveis que possam suprir a demanda mundial de forma sustentável é de extrema importância. O biodiesel produzido a partir de microalgas tem a capacidade de ser este substituto, no entanto, ainda são necessárias melhoras nos processos de produção de modo a torná-lo economicamente viável. Neste trabalho foi realizada a aplicação e comparação de um método de aprendizado por reforço com dois métodos de otimização clássicos em um modelo de crescimento de microalgas em biorreator contínuo. Os métodos de otimização utilizados foram: o método *Advantage Actor Critic*, um método de aprendizado por reforço com caráter exploratório que visa maximizar uma recompensa dada a ele dependendo do estado atingido por uma determinada ação; o algoritmo de busca local *Cobyla,*; e o algoritmo de busca global determinístico *Direct*. Os métodos tem como objetivo a maximização da produção de lipídeos no período de 300 horas. O método *Advantage Actor Critic* obteve valores de produção de lipídeos igual a 22,48 g, enquanto os métodos de otimização *Cobyla* e *Direct* obtiveram cerca de 200% da produção obtida pelo aprendizado por reforço. O tempo de operação do *Advantage Actor Critic* foi de 340,14 s, já os tempos para os métodos *Cobyla* e *Direct* foram de 70% e 50% do tempo do aprendizado por reforço respectivamente. Pela comparação dos resultados obtidos é possível observar que o método *Advantage Actor Critic* não obteve resultados que justificariam o seu uso frente aos otimizadores ao qual foi comparado.

Palavras-chave: *microalgas*, *aprendizado por reforço*, *advantage actor critic*, *otimização*

ABSTRACT

In face of growing oil prices and the environmental crisis that can't be ignored, the search for an alternative fuel source that can fully supply the worldwide demand is of extreme importance. Microalgae based biodiesel is a possible substitute, though optimization of the current microalgae production processes is still needed so that it can be economically viable. In this paper, a comparison is made between a reinforcement learning algorithm and two classic optimizers applied to a microalgae growth model in a continuous bioreactor. The methods used in this paper are the Advantage Actor Critic algorithm, an exploratory reinforcement learning method that maximizes a given reward based on the states reached taken a certain action, the Cobyla algorithm and the Direct algorithm. The three methods had the objective of maximizing the lipid production over the period of 300 hours. The Advantage Actor Critic method obtained the lipid production value of 22,48 g, while the cobyla and the Direct method obtained about 200% of the production obtained by the reinforcement learning method. The operating time of the Advantage Actor Critic method was 340,14 s, while the operating time for the Cobyla and Direct method were about 70% and 50% of the time used by the reinforcement learning respectively. By comparison of the tested methods, it can be determined that the Advantage Actor Critic method had a lower result that does not justify its use as opposed to the use of the other two methods.

Keywords: *microalgae*, *reinforcement learning*, *advantage actor critic*, *optimization*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 Introdução

Uma vez que não é possível alcançar sustentabilidade a longo prazo a partir de combustíveis fósseis, a busca por substitutos já vem acontecendo há décadas. Os substitutos renováveis "convencionais" são o etanol e o biodiesel, no entanto, ambos estão atrelados ao uso de biomassa, o que traz um novo problema a ser discutido. Uma vez que as fontes tradicionais de biomassa para a produção de combustíveis são produtos que envolvem o uso em larga escala de água e terra fértil, a substituição dos combustíveis à base de petróleo por essas fontes alternativas traria um agravamento de questões ambientais e sociais. A produção de lipídeos a partir de microalgas visando a produção de biodiesel é viável tecnicamente já que as microalgas apresenta alta produtividade aliada a uma grande concentração de lipídeos, sendo assim possível substituta à biomassa convencional utilizada na produção de biodiesel (Chisti, 2007). No entanto a viabilidade econômica dessa substituição é ainda bastante discutível.

Para a obtenção de maiores produtividades, técnicas de otimização devem ser aplicadas, desta forma, tanto otimizadores clássicos quanto métodos de aprendizado por reforço são técnicas válidas. Otimizadores tem como objetivo a maximização da função dada a partir de uma busca iterativa, podendo estar sujeitos aleatoriedades ou não, dentro dos valores permitidos para as variáveis do problema. O *Reinforcement Learning,* ou aprendizado por reforço, é uma forma de *Machine Learning* (aprendizado de máquina) que se diferencia dos outros por não necessitar de bancos de dados que suportem o seu aprendizado. Diferentemente de outros métodos de aprendizado de máquina, o aprendizado por reforço busca a maximização de uma dada recompensa para as ações tomadas por ele. O método trabalha de forma exploratória buscando diversas trajetórias durante seu período de aprendizado de modo a encontrar caminhos ótimos de operação com base nos retornos das recompensas geradas pelas ações tomadas por ele (Kotsiantis, 2007; S. Sutton e G. Barto, 2018). Aplicado à produção de microalgas, o aprendizado por reforço pode trazer novas possibilidades uma vez que torna possível a obtenção de novas trajetórias de processo que permitam a obtenção de melhores produtividades, melhorando a viabilidade dos processos.

Este trabalho visa fazer um comparativo de métodos de otimização clássicos com o aprendizado por reforço aplicado à produção de microalgas para obtenção de lipídeos. Para essa comparação foi feita a implementação de um algoritmo de aprendizado por reforço, com o método *Advantage Actor Critic* (Mnih et al., 2016), aplicado a um modelo matemático que descreve o crescimento de microalgas (Yoo et al., 2016), em um biorreator contínuo, assim como a otimização deste processo a partir do otimizador local *Cobyla* e do otimizador global *Direct* pela biblioteca *Scipy* (Virtanen et al., 2020) em *Python*.

Este trabalho foi dividido de forma convencional. O capítulo 2 traz a revisão bibliográfica referente aos materiais estudados neste trabalho, sendo eles a aprendizado por reforço e a produção de microalgas. O capítulo 3 descreve a

metodologia utilizada no trabalho, o modelo utilizado e os métodos de otimização implementados. O capítulo 4 apresenta os resultados obtidos para cada um dos métodos assim como os comparativos entre eles e o capítulo 5 as conclusões finais.

2 Revisão Bibliográfica

Este capítulo traz a revisão bibliográfica acerca dos métodos abordados neste trabalho. Juntamente é apresentada a fundamentação quanto aos temas de aprendizado por reforço e de produção de microalgas.

2.1 Reinforcement Learning

O r*einforcement learning* (aprendizado por reforço) é um tipo de *machine learning* (aprendizado de máquina) cuja fundamentação é a de imitar os processos de aprendizado de um ser vivo. Conforme se tomam decisões, existe a todo momento a ciência do ambiente e das consequências dessas ações. O aprendizado em geral não se dá necessariamente por uma parte externa ensinando o que fazer, mas muitas vezes a partir das observações feitas sobre o meio e as consequências geradas mostrando como seguir de modo a chegar aos objetivos traçados (Kotsiantis, 2007; S. Sutton e G. Barto, 2018).

O aprendizado por reforço se diferencia de outros métodos de aprendizado de máquina, como o aprendizado supervisionado e não supervisionado. O aprendizado supervisionado tem seu funcionamento baseado na análise de um banco de dados com parâmetros e as ações corretas a serem tomadas para seu treinamento. Neste caso o algoritmo tem como objetivo a generalização dessas situações que são então aplicadas a um conjunto com parâmetros conhecidos porém não classificados. Desta forma, esse tipo de aprendizado torna-se pouco prático em situações onde não existe um banco de dados para o treinamento, sendo esses muitos dos casos de fronteira do conhecimento (S. Sutton e G. Barto, 2018).

O aprendizado não supervisionado, por sua vez, se assemelha superficialmente com o aprendizado por reforço por não necessitar de um agente externo indicando as ações corretas a serem tomadas. No entanto, o aprendizado não supervisionado ainda utiliza uma base de dados aonde busca padrões estruturais buscando tendências nos parâmetros assim como comportamentos atípicos (Lesot, 2006; S. Sutton e G. Barto, 2018).

O aprendizado por reforço, por sua vez se baseia na existência de uma agente inserido em um ambiente com um objetivo claro. Esse agente então age com a intenção de chegar ao objetivo sem, no entanto, ter conhecimento certo sobre a estrutura do ambiente no qual está inserido. Desta forma as ações tomadas pelo agente se baseiam apenas nas recompensas obtidas como consequência das mesmas. Sendo assim, a partir dos resultados das suas ações o agente deve aprender a avaliar quais ações em quais situações trazem melhores resultados otimizando suas decisões com o decorrer do tempo.

De acordo com Sutton e Barto (2018) além do agente e o ambiente em que ele age, existem quatro elementos presentes no aprendizado por reforço: uma política, um valor de recompensa, uma função de valor e o modelo do ambiente. A política refere-se de certa forma ao análogo da resposta instintiva do agente, ela basicamente define como o agente se comporta dentro do seu ambiente correlacionando os eventuais estados do seu ambiente às ações a serem tomadas. A recompensa é o objetivo do agente, é um valor numérico dado ao final de cada ação tomada, podendo ser alto caso a ação resulte em um estado considerado como bom ou baixo no caso de um estado resultante considerado ruim, a definição de uma boa função de recompensa dependerá de um bom entendimento do processo. Sendo assim o

agente tem como objetivo maximizar essa recompensa ao longo do episódio, período de análise no qual o agente atua sobre o ambiente.

Atrelado ao valor de recompensa tem-se a função de valor, que tem como objetivo estimar recompensas futuras com base no estado atual do ambiente. Diferente das recompensas, que avaliam o resultado imediato de uma ação tomada pelo agente com base no estado consequente dessa ação, o valor corresponde ao resultado a longo prazo dessas ações com base nos estados futuros possíveis a partir dessa ação. Sendo assim, já que o objetivo final do agente é de maximizar a sua recompensa total, as suas ações se baseiam em encontrar estados de maior valor, não somente de maior recompensa, uma vez que esses trazem recompensas finais maiores independentemente das ações retornarem recompensas menores a outras ações partidas desse mesmo estado. “Recompensas são basicamente dadas diretamente pelo ambiente enquanto valores têm de ser estimados e reestimados a partir das observações feitas pelo agente no decorrer de todo o seu período de vida.” (S. Sutton e G. Barto, 2018).

O último elemento que se tem é o modelo do ambiente. Ele tem como objetivo simular o ambiente no qual o agente está imerso e calcular os novos estados com base nas ações tomadas pelo agente assim como o trabalho de dar recompensas ao agente com base nos estados atingidos a partir de suas ações com a partir dos estados originários.

Um importante desafio dos métodos de aprendizado por reforço é o equilíbrio entre a tendência exploratória e a tendência ambiciosa. Enquanto o algoritmo buscar pelo melhor resultado possível, com a maior recompensa total, ele estará priorizando estados e ações já conhecidas por darem a ele esses retornos. No entanto, para que esses estados de maior retorno sejam encontrados, é necessário que o agente tenha anteriormente decidido por ações não exploradas ao invés das conhecidas por terem maiores retornos. Isso demonstra que para um algoritmo de aprendizado por reforço obter os melhores resultados é necessário que haja um equilíbrio entre exploração e ambição, uma vez que se muito exploratório, não convergiria, e se muito ambicioso ficaria preso no primeiro máximo local encontrado (Grondman et al., 2012)

Diversos estudos vêm sendo realizados para avaliar a aplicabilidade do aprendizado por reforço a diversas áreas do conhecimento. Entre esses estudos pode-se exemplificar o uso do aprendizado por reforço na predição da dinâmica termoquímica na produção de plásticos com reforço de fibra de carbono (Szarski e Chauhan, 2021), na gestão da energia em casas inteligentes (Huang et al., 2021), no controle de sistemas hidrotérmicos (Saikia et al., 2011) e na integração com gêmeos digitais aplicados à cadeia logística (Abideen et al., 2021).

2.2 Método *Advantage Actor-Critic (A2C)*

A estrutura do método *Actor Critic* provém da união de duas famílias de métodos: o *Actor Only* e o *Critic Only*. Métodos *Actor Only* tem como base o uso de uma política parametrizada alimentada diretamente pela recompensa sem o uso de uma função de valor para estimar recompensas futuras. Otimizando a recompensa localmente e estimando um gradiente a cada passo, esses métodos atualizam os parâmetros da política na sua direção. A vantagem desses métodos é serem bastante convergentes, no entanto, as estimativas dos gradientes podem apresentar grande variância além de serem calculados sem o conhecimento de estimativas anteriores. Os métodos de *Critic Only*, por sua vez, utilizam a função de valor sem uma política,

sendo assim eles calculam uma política determinística com base em otimizações feitas sobre a função de valor. Esses métodos apresentam baixas eficiências além de não conseguirem atuar com espaços contínuos, discretizando-os por aproximações (Grondman et al., 2012).

O método *Actor Critic* visa unificar ambos os métodos *Actor Only* e *Critic Only* sendo assim composto por dois elementos que trabalham em sincronia: o ator e o crítico. O ator tem como função aprender a política do sistema enquanto o crítico aprende a função de valor. A principal vantagem desse sistema é que uma política pode ser melhor informada à tomar decisões quando alimentada com a função de valor ao invés das recompensas diretamente. Isso diminui a incerteza no aprendizado tornando-o, no entanto, mais complexo devido a ambos estarem sendo desenvolvidas simultaneamente, o que torna a tomada de decisões difícil até que se tenha valores mais congruentes sendo alimentados à política (Graesser e Keng, 2019). "Como os métodos de *Actor Only*, os métodos de *Actor Critic* possibilitam ações contínuas enquanto as grandes variâncias nos gradientes da política do *Actor Only* são balanceadas pela adição do crítico." (Grondman et al., 2012).

A Figura 1 esquematiza a estrutura de um método *Actor Critic*, nessa figura A representa a ação tomada, S representa o estado atual e R representa a recompensa obtida. O crítico processa a recompensa recebida e adapta a função de valor com base nela, ele então avalia a qualidade da política atual e após alguns passos de avaliação atualiza o ator. A linha tracejada demonstra que o crítico tem a função de atualizar o ator e a si mesmo. (Grondman et al., 2012).

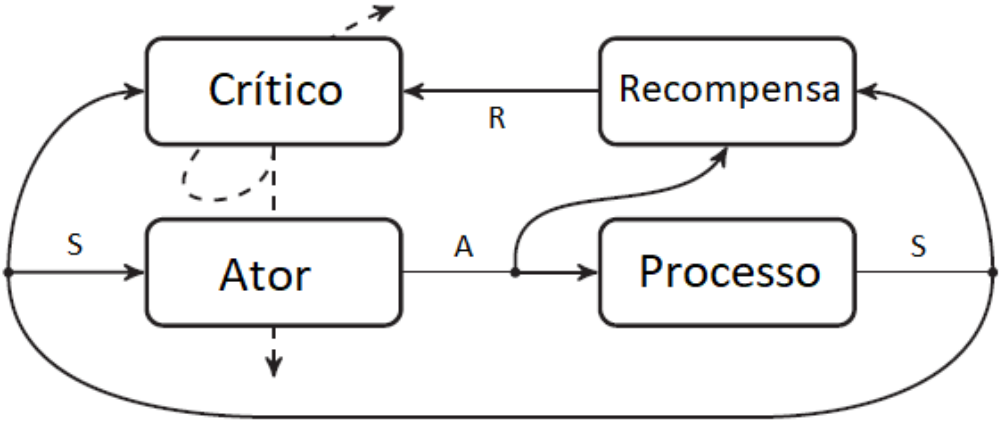

Figura 1: Esquema de um algoritmo *Actor Critic*. Adaptado de Grondman et al., 2012

O funcionamento básico de um método *Actor Critic* é com o ator contendo uma política parametrizada em θ, π_θ, e o crítico contendo uma função de valor parametrizada em ω, q_ω, onde θ e ω são os parâmetros que definem a distribuição das probabilidades sobre as ações na política e na função de valor respectivamente. A cada passo temos o estado atual S_t que é alimentado ao ator e ao crítico, o ator, com base no estado e na política, toma uma ação A_t, essa ação é alimentada ao crítico que então computa o valor gerado pela tomada dessa ação neste estado $q_\omega(St,A_t)$. A ação aplicada ao ambiente então retorna um novo estado S_{t+1} e uma recompensa R_{t+1}. O ator então atualiza os parâmetros θ da política (Eq. 1) com base na função de valor e com base nos novos parâmetros e estado toma uma nova ação A_{t+1}. O crítico então

atualiza os parâmetros ω da função de valor com base nas novas informações (Eq. 2) (Simonini e Sanseviero, 2022)

$$\Delta\theta = \alpha\nabla_{\theta}(log(\pi_{\theta}(S_t, A_t)))q_{\omega}(S_t, A_t) \quad (1)$$

$$\Delta\omega = \beta(R_{t+1}(S_t, A_t) + \gamma q_{\omega}(S_{t+1}, A_{t+1}) - q_{\omega}(S_t, A_t))\nabla_{\omega}q_{\omega}(S_t, A_t) \quad (2)$$

O método *Advantage Actor Critic* (Mnih et al., 2016) segue o funcionamento do *Actor Critic* porém com uma modificação. Ele utiliza de uma função de vantagem $A(S_t,A_t)$, uma modificação da a função de valor, como crítico com o intuito de minimizar a variância do método original (Eq. 3) onde $V(S_t,A_t)$ é o valor médio para o estado S_t. Essa função calcula o quão melhor é a tomada desta ação neste estado se comparado com o valor médio para o mesmo estado (Simonini e Sanseviero, 2022).

$$A(S_t, A_t) = q_{\omega}(S_t, A_t) - V(S_t) \quad (3)$$

2.3 Microalgas

O uso de combustíveis a base de petróleo não é algo sustentável a longo prazo, uma vez que suas fontes não são renováveis e trazem grandes danos ao ambientes atreladas ao seu uso. Logo novas fontes de combustíveis devem ser buscadas para contrabalançar os problemas dos combustíveis fósseis. O biodiesel é uma possível resposta para esse problema, porém o biodiesel é na sua maior parte produzido a partir de plantas que disputam o espaço de plantio de alimentos e utilizam de grandes quantidades de água no seu plantio e, portanto, não teriam capacidade de substituir por inteiro o uso de combustíveis fósseis. Sendo assim, microalgas parecem ser a única fonte de biodiesel capaz de suprir por inteira a necessidade por combustíveis (Chisti, 2007).

Microalgas são micro-organismos procariontes ou eucariontes fotossintéticos com necessidades simples de crescimento, como luz, água, CO_2, açúcares, nitrogênio, fósforo e potássio, com crescimento rápido e capazes de viver em ambientes severos devido a sua estrutura unicelular ou multicelular simples. Esses micro-organismos podem produzir grandes quantidades de lipídeos, carboidratos e proteínas em curtos períodos de tempo, devido ao seu curto ciclo de crescimento, que podem ser utilizados na produção combustíveis e outros produtos (Brennan e Owende, 2010; Mata et al., 2010).

A ideia de utilizar microalgas na produção de combustíveis não é algo novo, havendo pesquisas já na década de 70, com a primeira crise do petróleo. Porém, a crescente preocupação com o aquecimento global vinculado à queima de combustíveis fósseis, além do crescimento do preço do petróleo, tem trazido um maior interesse para essa possibilidade (Chisti, 2007; Mata et al., 2010).

De modo geral a produção de biodiesel a partir de microalgas apresenta diversas vantagens. Primeiramente o cultivo de microalgas é relativamente fácil, pode usar água imprópria para o consumo humano e nutrientes de fácil acesso. As diferentes espécies de microalgas podem ser adaptadas para viver em uma diversa gama de condições sendo possível escolhe-las com base nas condições locais. Devido a sua capacidade de viver em diversos ambientes e em condições severas, as microalgas podem ser cultivadas em áreas não propícias para outros cultivos, dessa forma não competindo com o plantio de alimentos ou criação de

animais. Elas possuem uma taxa de crescimento bastante alta e utilizam de espaços até duas ordens de grandezas menores do que as fontes de produção de biodiesel convencionais além de terem um alto teor de óleo na sua composição, que pode ser aumentado dependendo das condições do cultivo. O biodiesel produzido com microalgas performa tão bem quanto o diesel de petróleo além de reduzir emissões de particulados, hidrocarbonetos, CO e SOx (Mata et al., 2010).

2.3.1 Equipamentos para cultivo

Atualmente os processos de produção de microalgas fotoautotróficas se dividem em dois tipos: produção em lagoas abertas e a produção em fotobiorreatores fechados. O uso desses processos dependerá das condições climáticas do local assim como da espécie de microalga selecionada para o cultivo, tendo cada um suas vantagens e limitações. (Brennan e Owende, 2010)

A produção em lagoas abertas tem sido utilizada na produção de algas desde a década de 50 podendo ser separada em naturais e artificiais. O método mais utilizado na produção de microalgas em lagoas abertas artificiais é a produção em *raceways* que consiste em um corpo d'água fechado, em loop; e um aerador que funciona constantemente com o intuito de impedir a sedimentação. O fluxo é guiado nas curvas por defletores contidos no canal. No processo a lagoa é alimentada com uma corrente contendo nutrientes e microalgas à frente do misturador enquanto uma corrente de saída retira o material cultivado por detrás após percorrer o percurso completo. O CO_2 requerido para o crescimento das microalgas pode ser suprido diretamente pelo ar em contato com o meio de cultivo ou pela presença de aeradores submersos na lagoa (Brennan e Owende, 2010; Chisti, 2007).

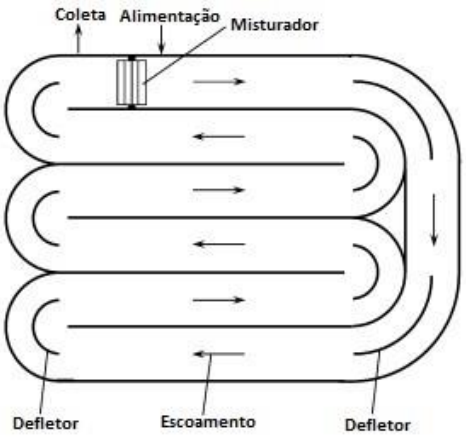

Figura 2: Visão aérea de uma *raceway. Adaptado de* Chisti, 2007

As vantagens da lagoa *raceway* é que ela tem um custo de implementação mais baixo do que o de um biorreator, tem processos de manutenção e limpeza mais fáceis, além de requerimentos energéticos menores. No entanto, as *raceways* possuem uma produtividade mais baixa do que os fotobiorreatores e maior facilidade de contaminação do meio de cultivo, uma vez que fica exposto ao ambiente. As *raceways* também utilizam de um grande espaço para o cultivo e têm baixa eficiência de mistura, iluminação e absorção de CO_2 (Brennan e Owende, 2010).

No cultivo em fotobiorreatores é possível o cultivo de uma mono cultura por períodos de tempo longos devido a menor possibilidade de contaminação. Alguns biorreatores tubulares consistem em uma série de tubos de pequeno diâmetro, usualmente de vidro ou plástico. Esse conjunto de tubos, coletores solar, fazem a captação da luz solar, quando não utilizada iluminação artificial. O cultivo é circulado de um reservatório para os coletores em fluxos turbulentos com o intuito de mitigar a sedimentação nos tubos como ilustrado na Figura 3 (Chisti, 2007).

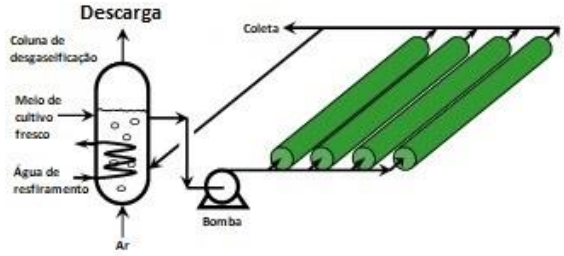

Figura 3: Fotobiorreator tubular com tubos dispostos na horizontal. Adaptado de Chisti, 2007

Na produção de microalgas em meios heterotróficos, tem-se o crescimento independente da fonte de luz. Nesses processos, o cultivo se dá em biorreatores de tanque agitado com a alimentação substratos ricos em carbono. Esses sistemas possibilitam um maior controle das condições de cultivo assim como menor custo de extração devido à maior densidade celular gerada (Brennan e Owende, 2010).

Em meios mixotróficos, tem-se o cultivo de espécies capazes de utilizar ambos processos metabólicos, desse modo o crescimento celular não é completamente dependente da luminosidade ou aos substratos alimentados, uma vez que ambos podem sustentar o crescimento celular. Meios mixotróficos apresentam uma diminuição da fotoinibição e da perda de biomassa nos períodos noturnos além de diminuir a necessidade de alimentação de substratos orgânicos. “Essas características inferem que a produção mixotrófica pode ser uma parte importante do processo de conversão de microalgas em biocombustíveis.” (Brennan e Owende, 2010).

2.3.2 Modelos de Crescimento

O crescimento de microalgas pode ser dividido em 6 etapas: a fase de adaptação, a fase de crescimento exponencial, a fase linear do crescimento, a fase de desaceleração do crescimento, a fase estacionária e a fase de morte, como ilustrado na Figura 4, onde a linha cheia descreve a concentração de microalgas enquanto a linha tracejada descreve a concentração de nutrientes no meio de cultivo. Na fase de adaptação há um atraso no crescimento devido a um ajuste fisiológico das células a um novo meio, esta fase é seguida pela fase de crescimento exponencial onde as células se dividem de forma exponencial em função do tempo. A fase de crescimento linear há um decréscimo na velocidade de crescimento devido à limitação da luminosidade no meio e se mantém um crescimento estável de biomassa até que a presença de nutrientes se torne um fator limitante. Na fase de desaceleração do crescimento a concentração de nutrientes no meio se torna limitante ao crescimento de forma que a taxa de crescimento diminui até se tornar igual à taxa de morte na fase estacionária, onde a população se mantém constante durante um período de tempo. Na fase de morte há um decrescimento do número de organismos, devido a fatores como

esgotamento de nutrientes, superaquecimento ou mudanças de pH, até que a população chegue a zero (Lee et al., 2015).

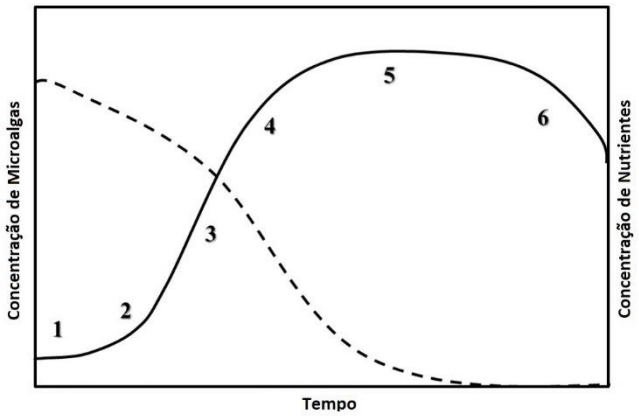

Figura 4: Ilustração das etapas de rescimento de microalgas. Adaptado de Lee et al., 2015

Até o momento, diversos modelos para determinar o crescimento de microalgas foram propostos. Lee et al., 2015, em sua revisão, separa os modelos de crescimento em três tipos, os que consideram um fator único de substrato, os que consideram apenas o fator luminosidade e os que consideram múltiplos fatores. O primeiro grupo pode ser subdividido em dois subgrupos: considerando que a taxa de crescimento depende da concentração de nutrientes no meio de cultivo e considerando que a taxa de crescimento depende da concentração de nutrientes no interior das células. O primeiro é exemplificado pelo modelo de Monod (Monod, 1949), porém não explica o crescimento em meios ausentes de nutrientes, uma vez que não considera a concentração intracelular. O modelo de Droop (Droop, 1968) é um exemplo do segundo subgrupo. Análises deste segundo subgrupo no entanto sofrem com a dificuldade tecnológica da medição das concentrações intracelulares (Lee et al., 2015).

Os modelos que consideram o fator luminosidade podem ser subdivididos em três subgrupos: considerando apenas a limitação da luminosidade, considerando a limitação da luminosidade associada à atenuação causada pelas células e considerando a limitação da luminosidade associada a atenuação causada pelas células e a fotoinibição. O modelo de Tamiya (H. Tamiya et al., 1953), Ogbonna (Ogbonna et al., 1995) e Steele (Steele, 1962) são exemplos dos três subgrupos respectivamente. (Lee et al., 2015)

O terceiro grupo de modelos são aqueles que consideram múltiplos fatores para o crescimento de microalgas. Esse grupo também pode ser dividido em dois subgrupos: os modelos de limite e os modelos multiplicativos. Os modelos de limite assumem que a taxa de crescimento de microalgas será afetado pelo fator mais limitante entre todos os analisados, desta forma seus equacionamentos finais são bastante similares aos modelos de fatores únicos. Os modelos multiplicativos por sua vez consideram que todas as limitações afetam a taxa crescimento simultaneamente. O modelo de Kunikane e Kaneko (Kunikane e Kaneko, 1984) é um exemplo deste subgrupo, no entanto existe uma certa dificuldade na aplicação deste modelo devido à sua estrutura complexa e à dificuldade na estimação de seus parâmetros. (Lee et al., 2015)

2.3.3 Otimização

A otimização dos parâmetros envolvidos no crescimento de microalgas é de enorme importância na viabilização econômica da sua produção. “Estudos quanto ao efeito de cada uma das variáveis independentes sobre a maximização do crescimento de microalgas [...] envolvem um número substancial de testes laboratoriais e, portanto, uma quantidade significativa de tempo, pessoas hábeis e outros recursos.” (Hossain et al., 2022). O uso de inteligência artificial, como redes neurais artificiais, são bastante populares na otimização de parâmetros devido a sua grande eficiência, acurácia e ampla gama de aplicações. (Hossain et al., 2022).

Diversos estudos quanto à otimização da produção de microalgas foram propostos na literatura como por exemplo a otimização do crescimento de microalgas em fotobiorreatores utilizando gás de combustão (He et al., 2012), a otimização do cultivo em lagoas abertas (Malek et al., 2016), a otimização da produção de biomassa e biofixação de CO_2 (Hossain et al., 2022) e a otimização de fotobiorreator de microalgas usando um modelo preditivo (Yoo et al., 2016). “Modelos das dinâmicas do cultivo de microalgas oferecem uma ferramenta de engenharia robusta para entendermos as respostas naturais porém complexas ao seu ambiente de crescimento e podem ajudar, se usados corretamente, a otimizar o cultivo de microalgas e aumentar a viabilidade econômica e sustentabilidade de sistemas de microalgas.” (Yoo et al., 2016). Deste modo o uso de modelos preditivos aliados a técnicas de otimização têm grande papel no desenvolvimento das tecnologias envolvendo microalgas assim como diversas outras áreas de pesquisa.

3 Materiais e Métodos

Na implementação deste trabalho foram utilizados um algoritmo de aprendizado por reforço e dois métodos de otimização dinâmica. Este capítulo apresenta e descreve o modelo matemático do crescimento de microalgas utilizado assim como os métodos utilizados na sua otimização.

3.1 Modelo

O modelo para o biorreator utilizado neste trabalho foi o proposto no artigo "*Optimization of microalgal photobioreactor system using model predictive control with experimental validation*" (Yoo et al., 2016), um modelo de múltiplos fatores de caráter multiplicativo. Esse modelo consiste em um sistema com múltiplas entradas e saídas descrito por seis equações de estado do sistema (Eq. 4 a 9) e um conjunto de 13 parâmetros cujos valores estão dispostos na Tabela 1 e 3 variáveis de entrada, sendo elas a vazão de alimentação de um substrato rico em nitrogênio (glicina) representada por f_1^i, de um substrato rico em carbono (glucose) representada por f_2^i e a iluminação incidida no meio de cultivo representada pro I.

$$\frac{dx}{dt} = \mu x + xD \tag{4}$$

$$\frac{dS_1}{dt} = -\rho x + S_1^i \frac{f_1^i}{V} - S_1 D \tag{5}$$

$$\frac{dS_2}{dt} = -\frac{1}{Y_{XS}}\mu x - \frac{1}{Y_{lS}}\pi x + S_2^i \frac{f_2^i}{V} - S_2 D \tag{6}$$

$$\frac{dN}{dt} = \rho x - \mu N - ND \tag{7}$$

$$\frac{dL}{dt} = \pi x - \mu L - LD \tag{8}$$

$$\frac{dV}{dt} = f_1^i + f_2^i - f_0 \tag{9}$$

Neste sistema, *x* corresponde à concentração de biomassa ativa (g/L), S_1 é a concentração do substrato rico em nitrogênio no meio (g/L), S_2 é a concentração do substrato rico em carbono no meio (g/L), N é a concentração intracelular de nitrogênio (g/L), L é a concentração intracelular de lipídeos (g/L), V é o volume. As concentrações das alimentações são conhecidas e têm valores de 10g/L e 200g/L representadas por S_1^i e S_2^i respectivamente. Para a manutenção do volume no biorreator foi dado o valor de f_0 como sendo a soma de f_1^i e f_2^i para a vazão de saída (f_0), tornando assim V constante, sendo esse valor igual a 2 litros.

O sistema de equações apresenta três taxas, a taxa de crescimento de biomassa ativa (μ), a taxa de absorção de nitrogênio (ρ) e a taxa de produção de lipídeos (π). Essas taxas são funções dos treze parâmetros do modelo assim como das concentrações, do volume, da razão de lipídeo acumulado (l), da razão de nitrogênio acumulado (q), da concentração total de biomassa (X) e da razão de diluição (D) (Eq. 10 a 16).

$$\mu = \mu_m\left(1 - \frac{q_0}{q}\right)\left(1 - \frac{l_0}{l}\right)\left(\frac{S_2}{K_{S_2} + S_2}\right)\left(\frac{I}{K_I + I}\right) \tag{10}$$

$$\rho = \rho_m\left(\frac{S_1}{S_1+K_{S_1}}\right)\left(\frac{q_m-q}{q_m-q_0}\right) \quad (11)$$

$$\pi = \pi_m\left(\frac{S_2}{K_\pi+S_2}\right)\left(1-\frac{N}{X}\right)\left(\frac{l_m}{1-l_m}\right) \quad (12)$$

$$q = \frac{N}{X} \quad (13)$$

$$l = \frac{L}{X} \quad (14)$$

$$D = \frac{f_1+f_2}{V} \quad (15)$$

$$X = (N + L + x) \quad (16)$$

Tabela 1: Parâmetros do modelo de crescimento de microalgas conforme o estudo de Yoo et al., 2016

Parâmetros	Valores	Unidades
Taxa máxima de crescimento, μ_m	0,0418	1/h
Taxa máxima de produção de lipídeos, π_m	0,0762	1/h
Razão mínima de nitrogênio acumulado para o crescimento, q_0	0,0196	g/g
Razão mínima de lipídeo acumulado para o crescimento, l_0	0,0006	g/g
Razão máxima de nitrogênio acumulado para absorção, q_m	0,2109	g/g
Razão máxima de lipídeo acumulado para a produção de lipídeos, l_m	0,6995	g/g
Constante de meia saturação da glicina para a absorção, K_{S1}	0,5793	g/L
Constante de meia saturação da glucose para o crescimento, K_{S2}	0,1002	g/L
Constante de meia saturação da luminosidade para o crescimento, K_I	66,5337	µmol/m²s
Constante de meia saturação da produção de lipídeos, K_π	12,5596	g/L
Máxima taxa de absorção, ρ_m	0,1197	1/h
Coeficiente de rendimento de glucose em biomassa, Y_{XS}	0,9597	g/g
Coeficiente de rendimento de glucose em lipídeo, Y_{lX}	0,1908	g/g

Fonte: Adaptado de (Yoo et al., 2016)

3.2 Aprendizado por Reforço

Para iniciar o processo de otimização pelo método de aprendizado por reforço é necessário estruturar o ambiente e o algoritmo de aprendizado. No ambiente estão contemplados o

espaço de ações possíveis de serem tomadas pelo agente do aprendizado por reforço assim como a definição dos estados atingidos para cada ação tomada. O algoritmo contempla a política do método assim como os seus parâmetros, como a taxa de aprendizado e configuração da rede neural.

O ambiente foi construído na biblioteca *Open AI Gym* do *Python*. O espaço de ações foi descrito como um conjunto de três ações distintas a serem tomadas variando os valores de f1, f2, e I a cada passo de dez horas durante um episódio de trezentas horas, sendo assim trinta e uma ações, contando com a decisão inicial para cada uma das variáveis de entrada, totalizando um total de noventa e três ações por episódio. Essas ações podem variar entre valores discretos de zero a vinte que são corrigidos por fatores de conversão para cada uma das variáveis e somados aos valores anteriores das variáveis de entrada (Eq. 17 a 19).

$$f_1^i = f_1^{i-1} + (A - 10)\frac{0{,}01}{100} \tag{17}$$

$$f_2^i = f_2^{i-1} + (A - 10)\frac{0{,}01}{100} \tag{18}$$

$$\mathrm{I}^i = \mathrm{I}^{i-1} + (A - 10)\frac{740}{100} \tag{19}$$

Dentro do ambiente também estão descritas as faixas de valores possíveis para cada variável. Os valores mínimos e máximos para cada variável decididos foram de 0 a 0,01 para f_1 e f_2 e de 10 a 740 para I, seguindo os valores estipulados no trabalho original (Yoo et al., 2016). As ações tomadas pelo agente foram calibradas de modo que ele pudesse variar em até 10% da variação máxima de cada variável em uma única ação. Foram também estipulados limites máximos para os valores de S_1 e S_2, sendo esses 10g/L e 200g/L respectivamente, o que corresponde às concentrações das vazões de entrada.

As vazões de entrada de glicina (f_1), glucose (f_2) e incidência luminosa (I) foram 0,0016 L/h, 0,0018 L/h e 500 µmol/m²s respectivamente como ponto de partida para o método. Para o estado inicial, foram utilizados os valores de 0,1 g/L para a concentração de biomassa ativa (*x*), 5 g/L para a concentração de glicina no meio reacional (S_1), 100 g/L para a concentração de glucose no meio reacional (S_2), 0,02 g/L para a concentração de nitrogênio intracelular (N) e 0,001 g/L para a concentração de lipídeos intracelulares (L).

Como o objetivo do estudo é a maximização da produção de lipídeos, a recompensa foi calculada em base dos valores de concentração de lipídeos no meio a cada passo do episódio. Desse modo foi decidido o valor final da recompensa de cada ação seria o somatório das concentrações intermediárias obtidas no período, multiplicados pelos valores da vazão de saída no período multiplicados pelo tempo, em horas, entre cada ponto, neste caso 0,1 h. Como ilustrado na Figura 5 o valor final da recompensa é a integral da curva da taxa de produção de lipídeos que foi resolvida discretamente a partir dos pontos tomados. Os valores de concentração L foram registrados em cem pontos para cada passo de dez horas totalizando um total de três mil pontos considerados no período final do episódio de 300 horas. A recompensa final (R) do episódio é então o somatório das recompensas retornadas a cada passo (Eq. 20).

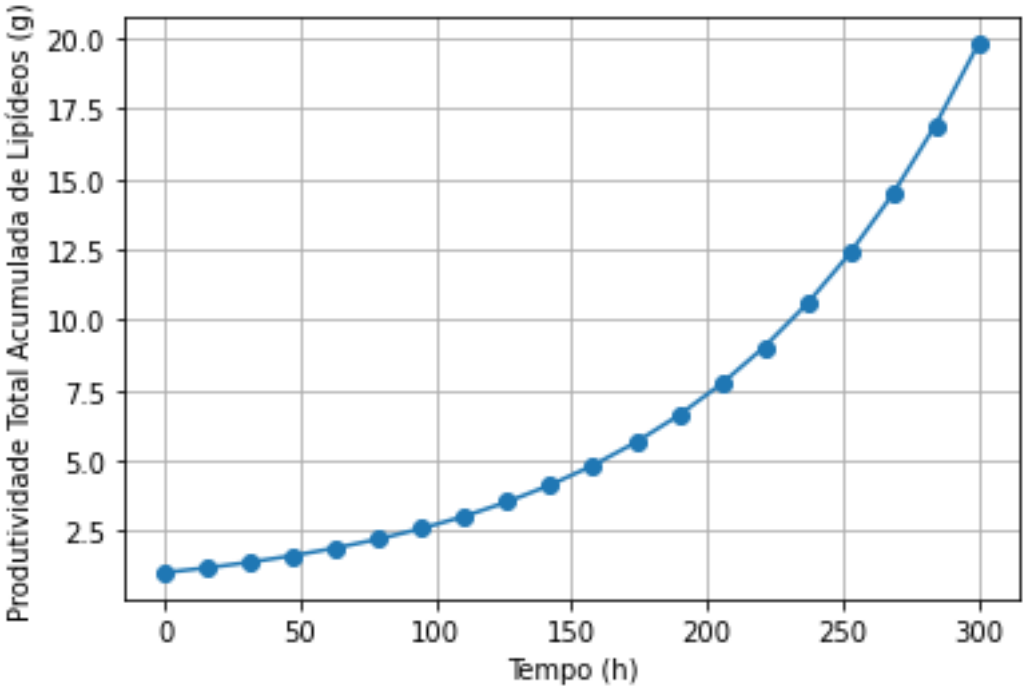

Figura 5: Ilustração da recompensa total obtida pelo algoritmo de aprendizado por reforço

$$R = \sum_{n=0}^{N}(\sum_{i=0}((L_{i,n})(0{,}1f_{0_n}))) \quad (20)$$

onde i representa os pontos intermediários do passo atual, n é o passo atual e N é o número total de passos em um episódio.

O algoritmo selecionado para este trabalho foi o *advantage actor critic,* uma versão síncrona do *Asynchronous Advantage Actor Critic* (A3C) proposto no artigo de Mnih et al. (2016). O algoritmo foi implementado a partir da biblioteca *stable baselines 3* em *Python* com política *MlpPolicy,* a decisão quanto ao uso desta biblioteca foi devido a sua simplicidade de implementação assim como a ampla documentação disponível. Para o aprendizado, foram analisados os seguintes pontos: os valores da recompensa final e o tempo de execução do algoritmo. Durante os treinamentos foi fixado um mesmo valor para a *seed*, igual a 13, de modo a dar repetibilidade e facilitar as comparações entre as configurações utilizadas.

Inicialmente foi decidido por analisar o aprendizado com duas redes neurais de 64 neurônios, com o número de episódios sendo 100000 e variando a taxa de aprendizado entre os valores de 0,0005, 0,001, 0.005 e 0,01 para verificar a influência deste sobre a produtividade final. Posteriormente foram realizados novos testes desta vez variando o tempo de aprendizado entre períodos de 100000, 200000 e 400000 episódios com uma taxa de aprendizado de 0,005 para avaliar se o aumento do número de episódios teria grande influência sobre o resultado final. Por fim foi analisado se o aumento da rede neural utilizada teria grandes efeitos sobre a recompensa, utilizando duas redes de 64 neurônios, duas redes de 128 neurônios e três redes neurais com 128 neurônios. A decisão quanto a configuração das redes neurais foi com base na configuração padrão, utilizada no primeiro teste, sendo a primeira variação dobrar o número de neurônios nas redes e a segunda adicionar uma nova rede com o mesmo número de neurônios.

3.3 Otimização

Com o intuito de comparação do método selecionado para este trabalho foi decidido por uma otimização dinâmica do modelo. Para a implementação dos algoritmos de otimização, três variáveis manipuladas foram discretizadas em 31 intervalos, para um período de 300 horas de processo, contemplando então 93 variáveis de decisão. Essas variáveis foram agrupadas em uma lista e convertidas em valores de f_1, f_2 e I pelo cálculo apresentado nas Equações 17, 18 e 19 respectivamente, onde neste caso os valores de A representam as variáveis de decisão.

Para o processo de otimização foram dados como valores iniciais um conjunto zeros, com exceção dos primeiros valores aos quais foram atribuídos os valores de 0,0016L/h, 0,0018L/h e 500 µmol/m²s, para f_1, f_2, e I respectivamente, de modo a facilitar a inicialização do método. Diferentemente do método de aprendizado por reforço, os métodos de otimização clássicos estavam livres para definir os valores iniciais das variáveis manipuladas. Os limites das variáveis de decisão foram restringidos a valores entre 0 e 0,01 para f_1 e f_2 e 10 e 750 para I. Também foram decididos os valores iniciais da concentração de biomassa ativa, da concentração de glicina no meio reacional, da concentração de glucose no meio reacional, da concentração de nitrogênio intracelular e da concentração de lipídeos intracelulares, sendo esses 0,1g/L, 5g/L, 100g/L, 0,02g/L e 0,001g/L respectivamente. Como função objetivo para o otimizador foi calculada a produção total de lipídeos durante o período de 300 horas (Eq. 20), sendo o objetivo do otimizador minimizar o resultado do inverso dessa função (Eq. 21).

$$g(f_1, f_2, I) = -\sum_{n=0}^{N}\left(\sum_{i=0}\left(\left(L_{i,n}\right)\left(0{,}1 f_{0_n}\right)\right)\right) \tag{21}$$

A otimização foi implementada a partir da biblioteca *scipy* (Virtanen et al., 2020) utilizando um otimizador local e outro global. O otimizador local selecionado foi o método *Cobyla*, um método sem derivada, para problemas com restrições, que funciona com base na resolução de diversos problemas lineares aproximados em sequência de modo que cada uma das iterações tem como objetivo melhorar a aproximação linear utilizada na iteração que a sucede, até que o passo se torne pequeno o suficiente para que se considere a solução como ótima localmente. O método de otimização global utilizado foi o *Direct* ou *Dividing Rectangles,* um otimizador global determinístico que faz a minimização com base na amostragem de soluções em potencial no espaço de busca, ele funciona com base no particionamento do espaço de buscas que potencialmente possa conter o ótimo global, o algoritmo segue buscando espaços em potencial e os particionando até que se encontre o ótimo global.

4 Resultados

Os resultados obtidos dos experimentos com o método *Actor Critic* não apresentaram vantagens sobre os outros métodos uma vez que obteve valores mais baixos do que os outros métodos além de ter um tempo de operação maior do que ambos. O método *Direct* foi o que apresentou melhores resultados entre os dois métodos de otimização utilizados como comparação, algo esperado uma vez que é um método de otimização global desta forma sofrendo menos com mínimos locais distribuídos pelo espaço de análise. Na Figura 6 é apresentada uma síntese dos resultados obtidos para cada um dos métodos comparados.

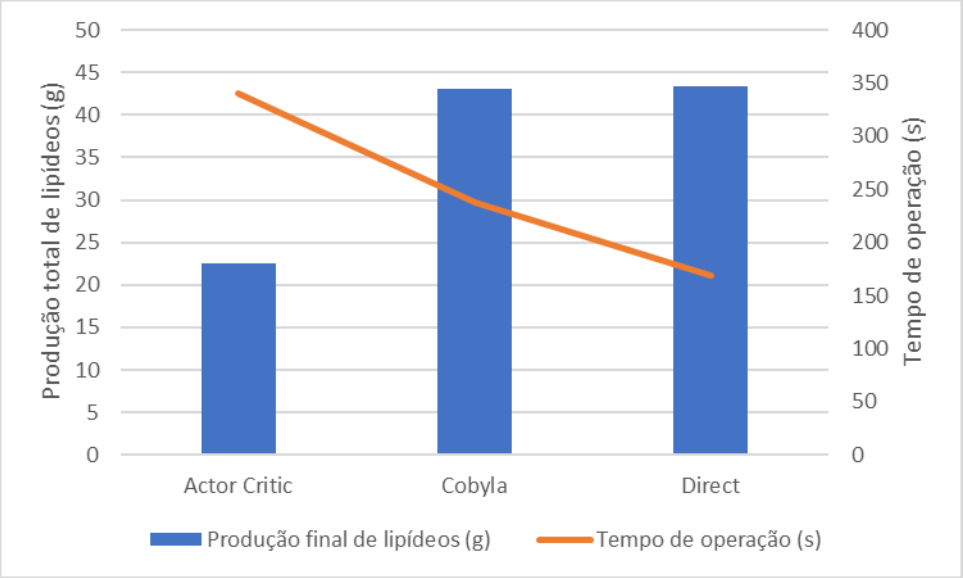

Figura 6: Resultados finais das otimizações

4.1 Método *Cobyla*

O método *Cobyla*, assim como o *Direct*, foi utilizado como ferramenta de comparação para verificar a eficiência do método de aprendizado por reforço na otimização do processo de produção de microalgas. Como informado anteriormente este modelo apresentou resultados melhores do que os obtidos pelo *Actor Critic,* alcançando uma valor de produção de lipídeos de 43,12 g em 237,62 s de operação. A Figura 7 apresenta os valores de f_1 tomados pelo agente durante o período de 300 horas, a Figura 8 apresenta as ações tomadas sobre a variável f2 e a Figura 9 apresenta as ações tomadas sobre a variável I. A Figura 10 apresenta a variação da concentração intracelular de lipídeos no mesmo período, sendo esses valores os diretamente proporcionais à recompensa finais obtidas.

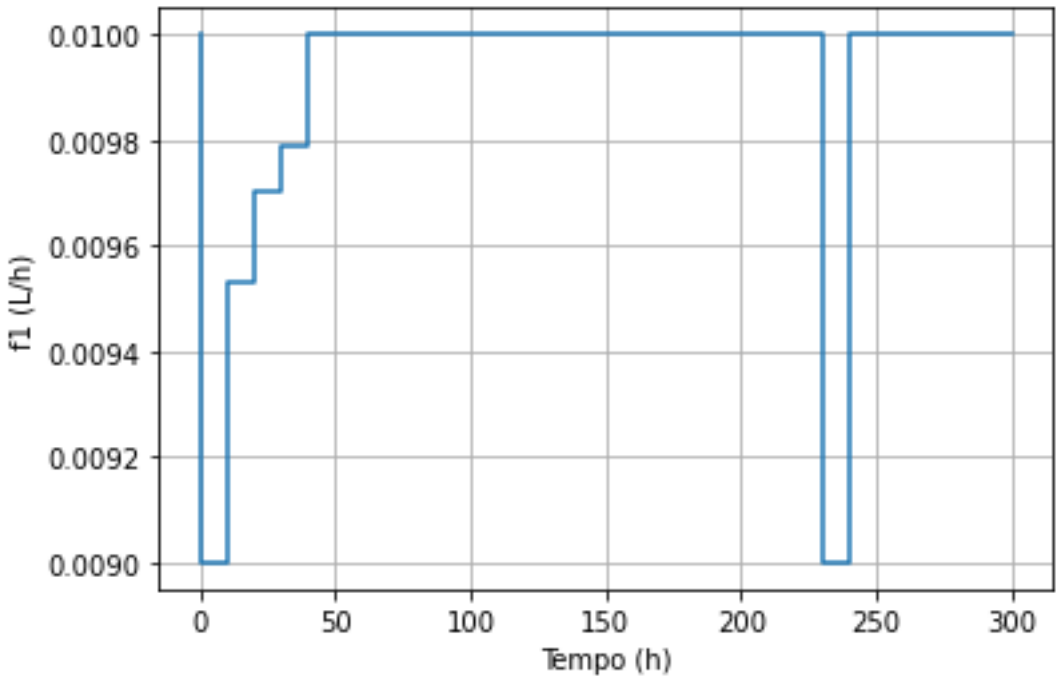

Figura 7: Variação da vazão de entrada de glicina em função do tempo de operação para o método *Cobyla.*

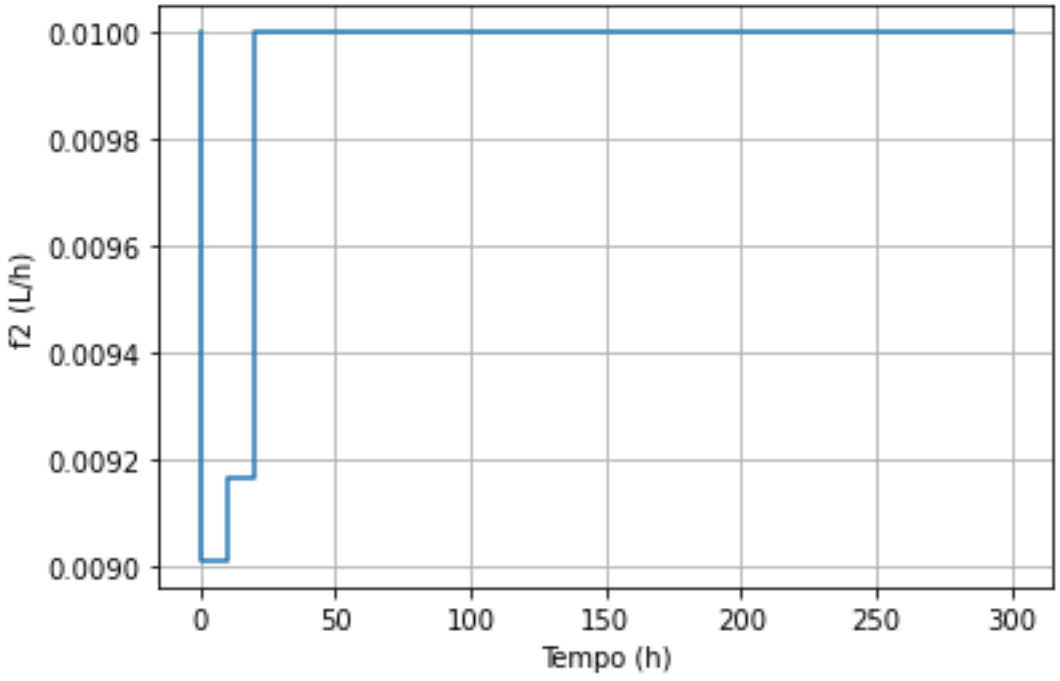

Figura 8: Variação da vazão de entrada de glucose em função do tempo de operação para o método *Cobyla.*

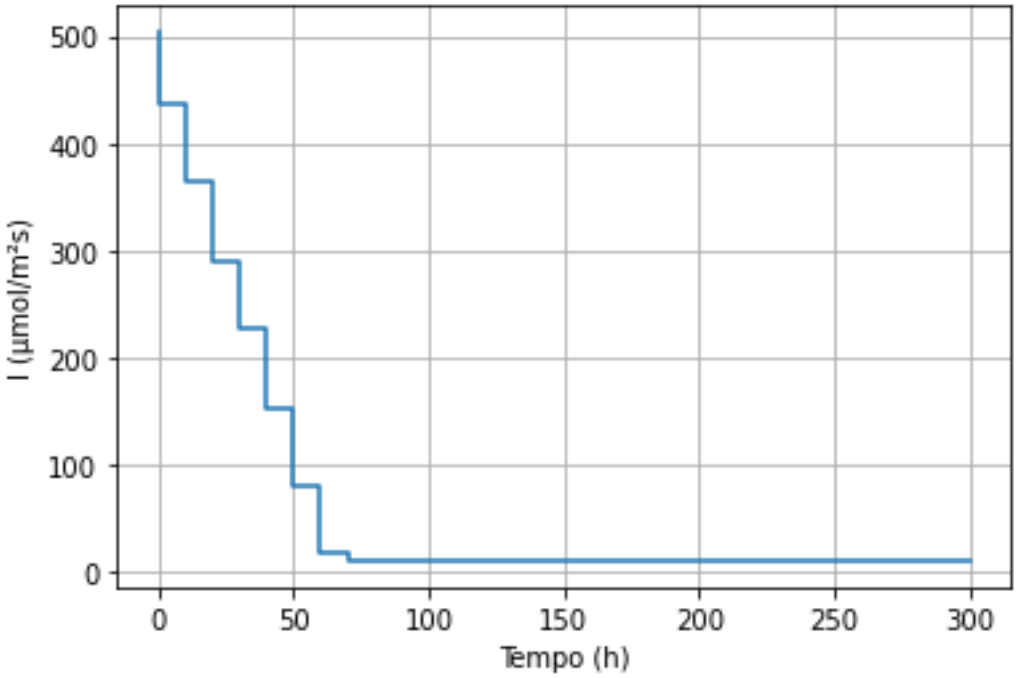

Figura 9: Variação da incidência luminosa em função do tempo de operação para o método *Cobyla.*

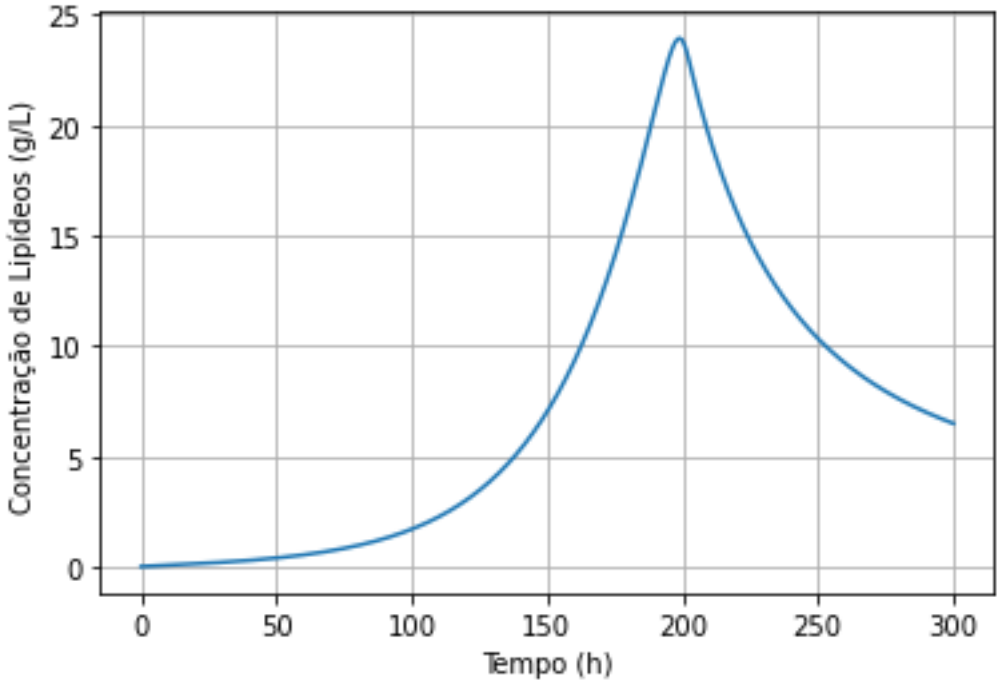

Figura 10: Variação da concentração de lipídeos intracelular no período de teste para o método *Cobyla*.

Pode ser observado nos gráficos que apresentam as variáveis manipuladas que, apesar de certa variação no período estudado, os valores tenderam a se manter próximos ao máximo de cada uma das variáveis. Isso pode ser esperado pelo fato da função objetivo ser a produção total no período que é função da variável de saída, que por sua vez está atrelada aos valores

das vazões de entrada f1 e f2 servindo como controle de inventário. Já a incidência de luz pode ter se mantido nesses valores pelo fato do modelo não considerar os efeitos da fotoinibição.

4.2 Método *Direct*

O método *Direct* foi o que apresentou os melhores resultados frente aos três testados neste trabalho. Este método obteve o maior rendimento, apesar de próximo ao valor alcançado pelo método *Cobyla*, com uma produção de 43,31g de lipídeos no período de trezentas horas, assim como apresentou o menor tempo de operação entre os três métodos, 168,7 segundos, cerca de 30% menor do que o tempo de execução do método *Cobyla* e um pouco menos que a metade do tempo do *Actor Critic*. As ações tomadas pelo método podem ser observadas nas Figuras 11, 12 e 13. A Figura 11 apresenta os valores da variável f_1 durante o período de 300 horas, a Figura 12 apresenta os valores da variável f_2 e a Figura 13 apresenta os valores da variável I. Os valores da concentração de lipídeos m função do tempo estão representados na Figura 14. Assim como para o método *Cobyla* é possível observar que os valores das variáveis f_1 e f_2 variam no entorno de seus valores máximos.

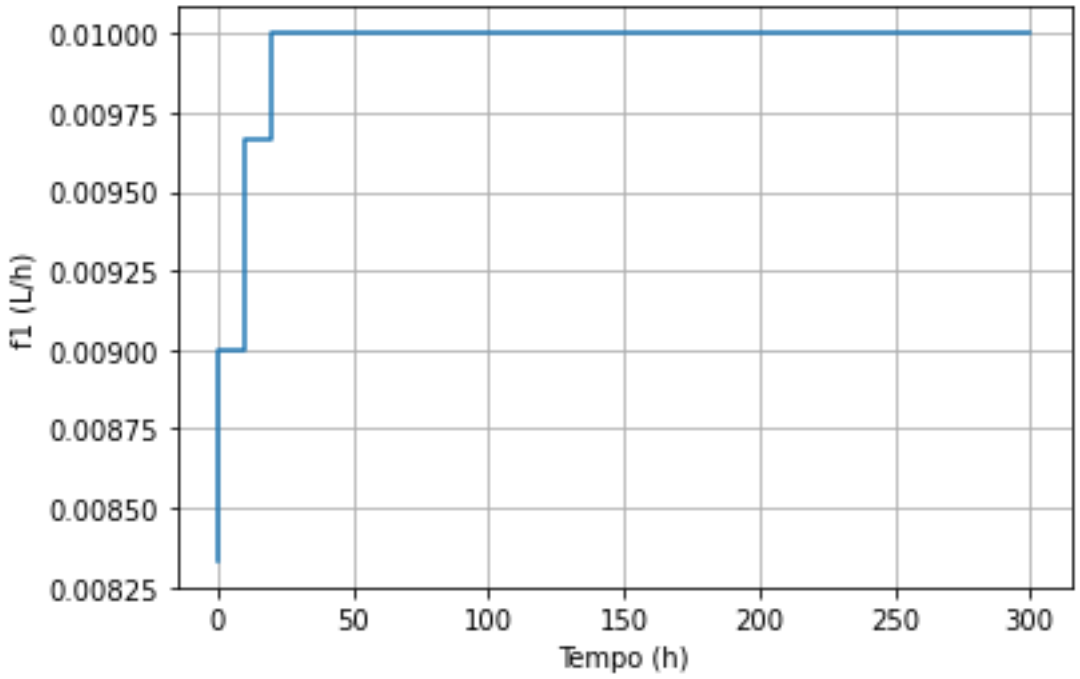

Figura 11: Variação da vazão de entrada de glicina em função do tempo de operação para o método *Direct*.

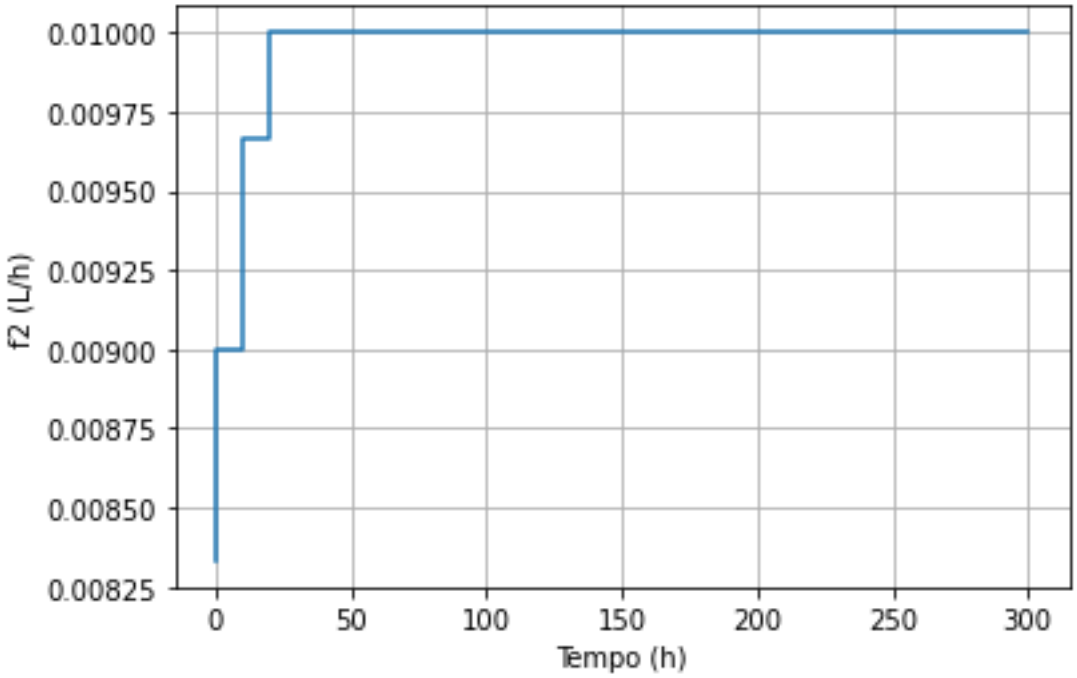

Figura 12: Variação da vazão de entrada de glucose em função do tempo de operação para o método *Direct*.

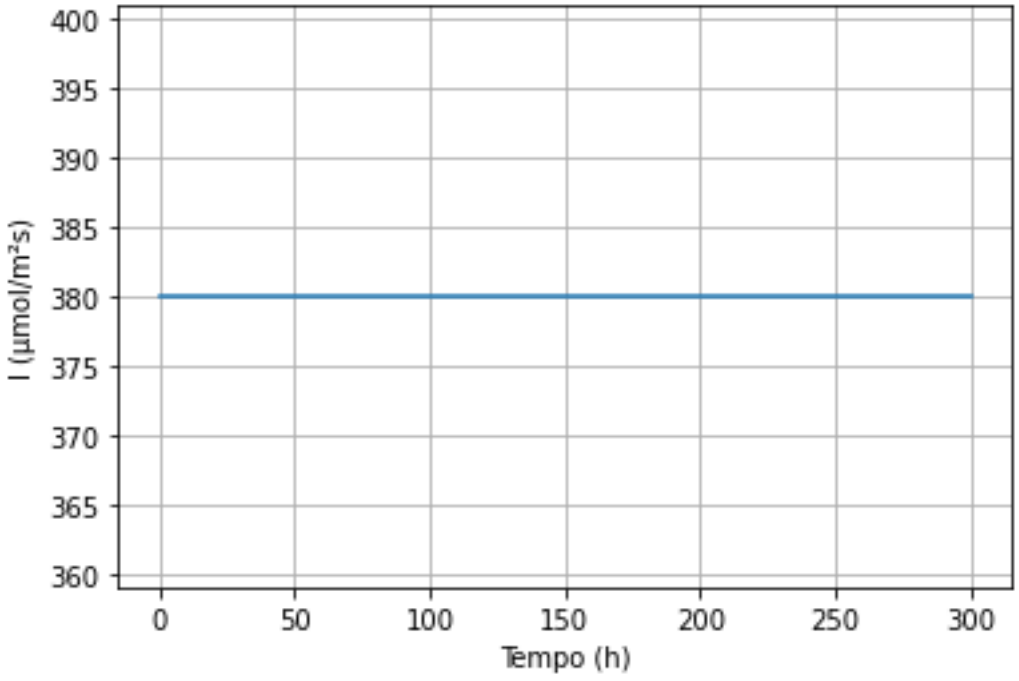

Figura 13: Variação da incidência luminosa em função do tempo de operação para o método *Direct.*

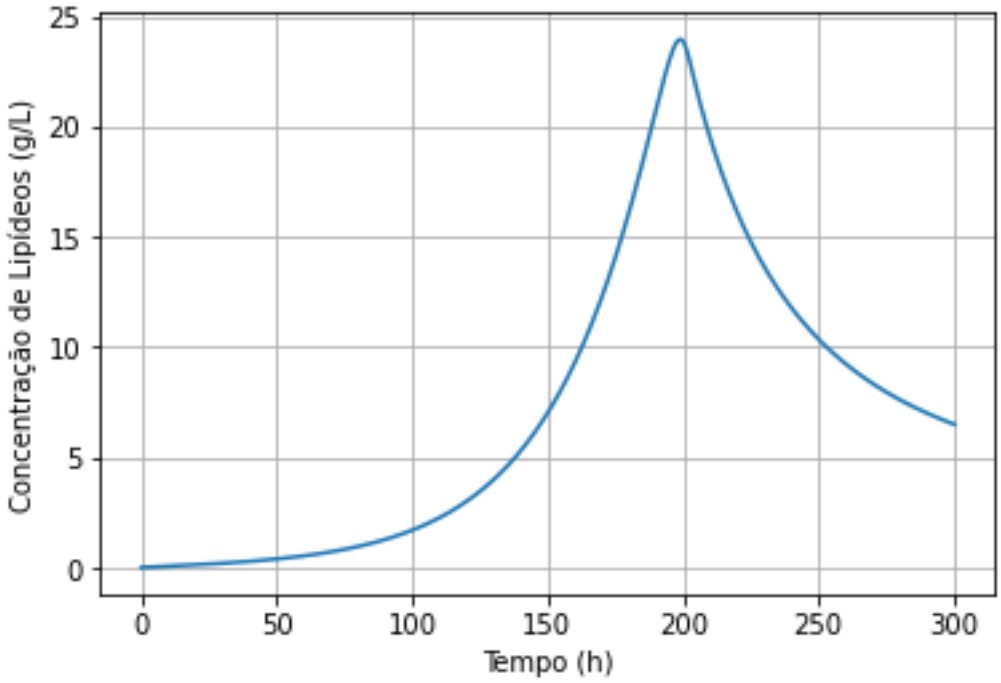

Figura 14: Variação da concentração de lipídeos intracelular no período de teste para o método *Direct.*

4.3 Método *Actor Critic*

Durante os períodos de treinamento foi feita uma avaliação da recompensa a cada 10000 episódios para avaliar a política de ações do método. Todos os resultados apresentados são uma média de 11 valores de recompensa obtidos com base nas melhores políticas alcançadas durante o período de treinamento.

Na primeira avaliação, o método *Actor Critic* operou inicialmente em um período de aprendizado de 100000 episódios com duas redes neurais de 64 neurônios e uma taxa de aprendizado variando entre os valores de 0,0005, 0,001, 0,005, 0,01. Os resultados desta análise se encontram na tabela 2. Uma vez que a taxa de aprendizado de 0,005 obteve os maiores valores para a recompensa, ela foi utilizada para realizar as análises sobre a influência do número de episódios sobre a recompensa final.

Tabela 2: Síntese dos resultados da variação da taxa de aprendisado sobre a recompensa final

	0,0005	0,001	0,005	0,01
Tempo de operação (s)	429,45	344,10	340,14	326,64
Produção final de lipídeos (g)	14,29	12,72	22,48	12,59

Na segunda análise, que contempla o número de episódios para o treinamento foram feitos três testes com 100000, 200000 e 400000 episódios. Os valores resultantes destes testes estão descritos na tabela 3. Como pode ser observado, o

aumento do número de episódios não trouxe variação na recompensa obtida, demonstrando que o método encontra a sua melhor política em um período abaixo dos 100000 episódios.

A última etapa de testes consistiu em executar o treinamento variando as redes neurais para duas redes de 128 neurônios e três redes de 128 neurônios para avaliar se esse aumento causaria uma variação significativa na recompensa final adquirida. Após os testes foi observado que o valor de recompensa se manteve constante apesar das mudanças na configuração da rede neural. Esses testes foram realizados em 100000 episódios com a taxa de aprendizado em 0,005. Os resultados estão descritos na tabela 4.

Tabela 3: Síntese dos resultados da variação o número de episódios sobre a recompensa final

	100000	200000	400000
Tempo de operação (s)	340,14	654,38	1278,78
Produção final de lipídeos (g)	22,48	22,48	22,48

Tabela 4: Síntese dos resultados da variação do tamanho da rede neural sobre a recompensa final

	2 redes de 64	2 redes de 128	3 redes de 128
Tempo de operação (s)	340,14	329,62	328,06
Produção final de lipídeos (g)	22,48	22,48	22,48

O objetivo da aplicação do método era o de encontrar um mapa de ações ótimos durante o período de análise que maximizasse a produção final de lipídeos do reator. No entanto os melhores resultados do método não se mostraram bons concorrentes se comparado aos métodos de otimização dinâmica, necessitando de períodos de treinamento mais longos que o de execução dos outros métodos, cerca de 100% e 40% mais do que os otimizadores *Direct* e *Cobyla* respectivamente, e obtendo resultados abaixo dos mesmos, cerca de 50% dos valores obtidos por ambos os métodos.

Os resultados finais do método, utilizando a configuração de 0,005 para a taxa de aprendizado, em 100000 episódios e com duas redes neurais de 64 neurônios, podem ser observados nas Figuras 15 a 18. A Figura 15 apresenta os valores da variável f_1 durante o período de 300 horas, a Figura 16 apresenta os valores da variável f_2 e a Figura 17 apresenta os valores da variável I. Na Figura 18 estão representados os valores da concentração intracelular de lipídeos ao longo do período de análise.

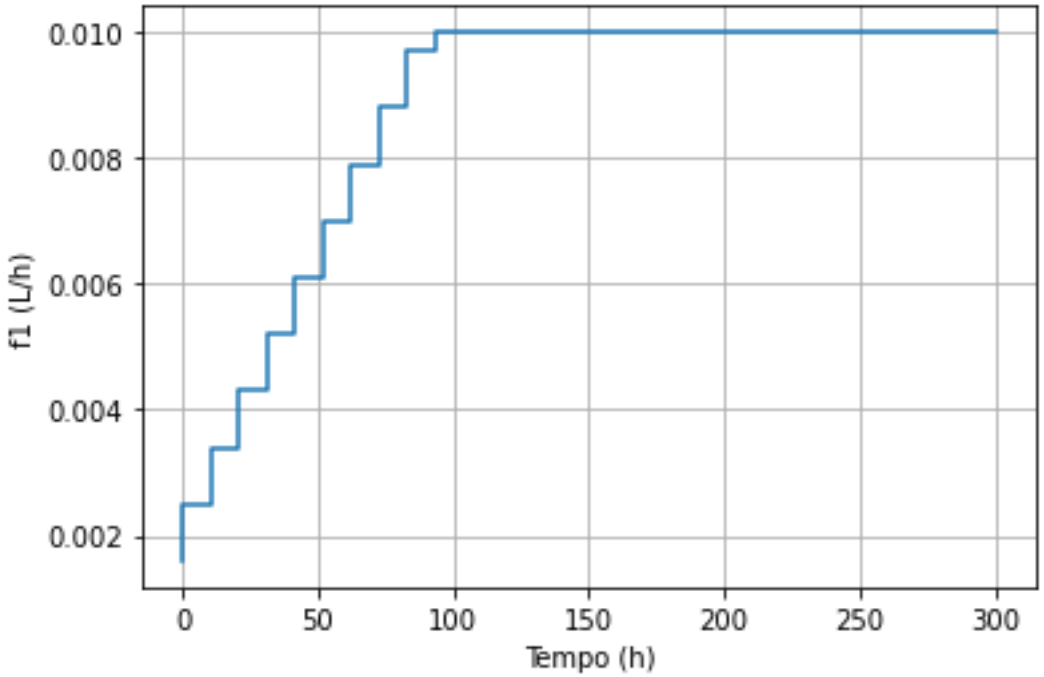

Figura 15: Variação da vazão de entrada de glicina em função do tempo de operação para o método *Actor Critic*.

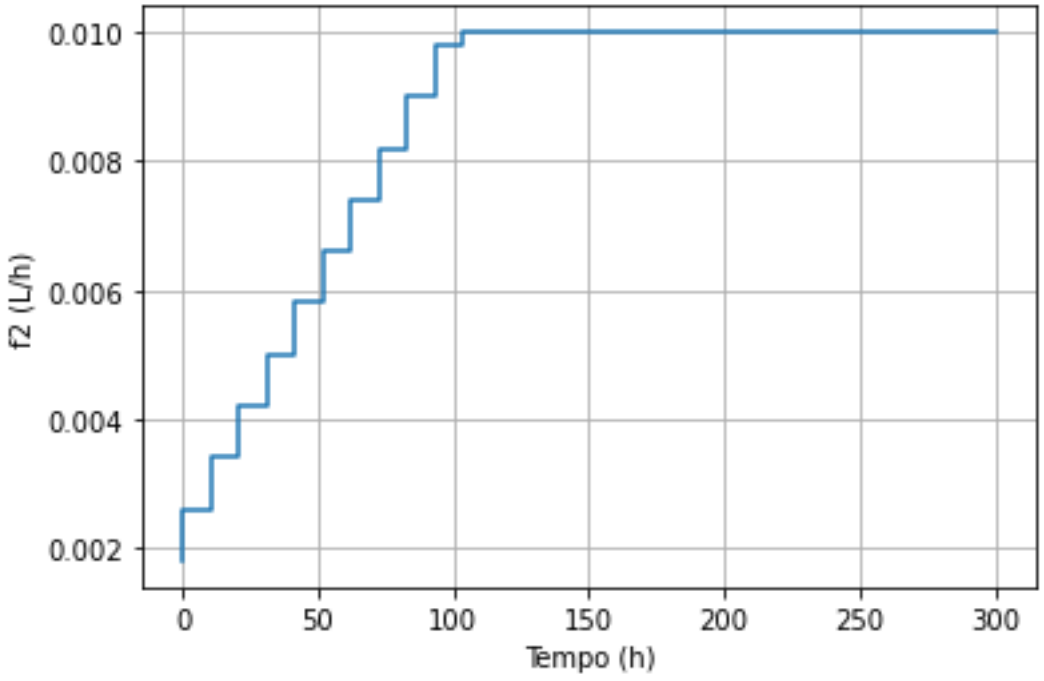

Figura 16: Variação da vazão de entrada de glucose em função do tempo de operação para o método *Actor Critic*.

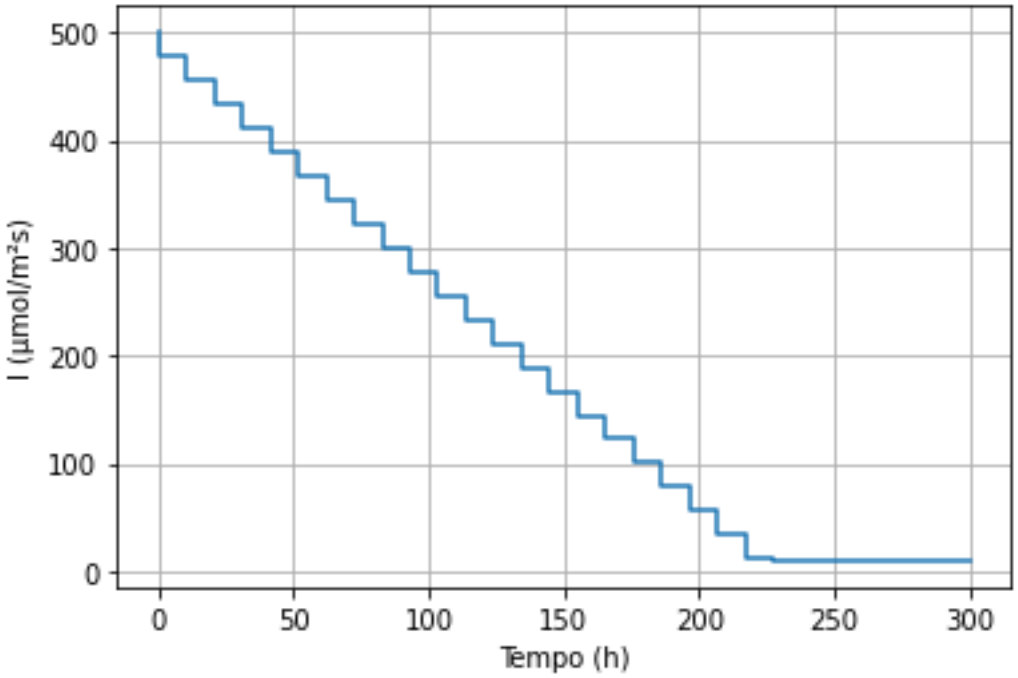

Figura 17: Variação da incidência luminosa em função do tempo de operação para o método *Actor Critic*.

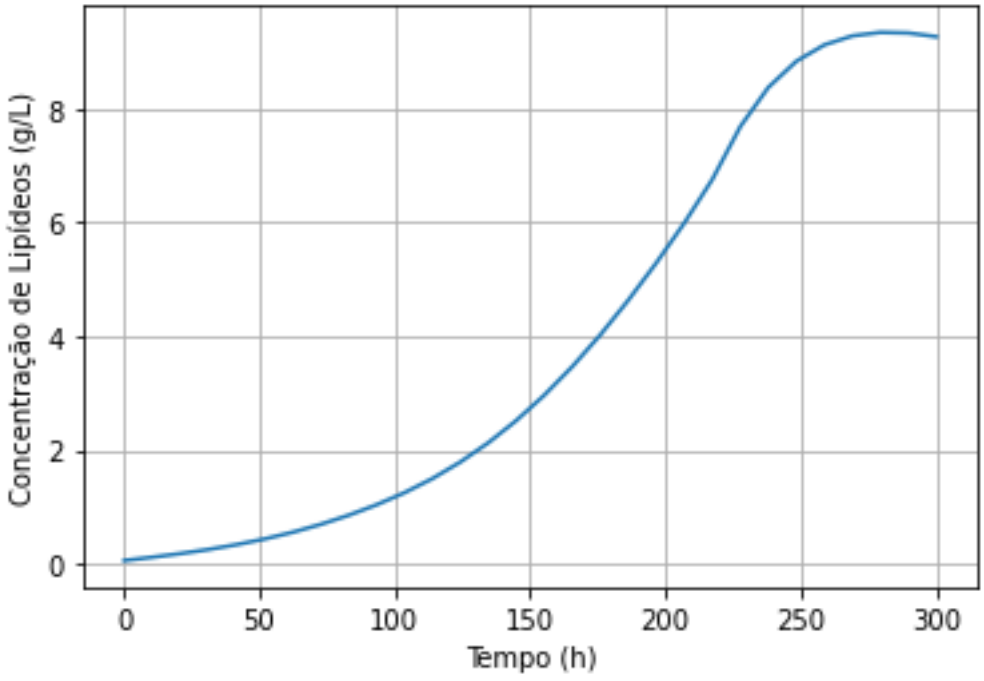

Figura 18: Variação da concentração de lipídeos intracelular no período de teste para o método *Actor Critic*.

4.4 Comparativo da Literatura

Atualmente não há trabalhos na literatura que estudem a aplicação de aprendizado por reforço na produção de microalgas, porém há diversos estudos com aplicações em outras áreas. No seu estudo, Petsagkourakis et al., 2020 traz um comparativo de método de

aprendizado por reforço adaptado a bioprocessos complexos com controladores preditivos de modelos não-lineares com resultados melhores do que os apresentados pelo controlador preditivo. Powell et al., 2020 faz a comparação de otimizadores em tempo real baseados em aprendizado por reforço e otimização não-linear, apesar de obter resultados cerca de 50% menores que o otimizador com base em otimizador não-linear, isso ainda é um resultado promissor uma vez que o mesmo opera com um modelo quase perfeito do processo assim como o fato de o otimizador baseado em aprendizado por reforço toma cerca de 13% do tempo para processar o período de análise em comparação com o otimizador não linear. Em seu trabalho de conclusão de curso em engenharia química Kucyk, 2021 faz um comparativo de aprendizado por reforço e otimizadores dinâmicos clássicos aplicados a otimização de poços de elevação de petróleo com resultados para os valores de objetivo competitivos se comparados aos dos otimizadores clássicos porém em tempos bastante maiores, chegando a cerca de 120 vezes o tempo de otimização para o melhor método estudado.

Estudos quanto ao uso de métodos de otimização dinâmica aplicados à produção de microalgas já são mais comuns, De-Luca et al., 2018 em seu estudo apresentou uma metodologia para a otimização da produção de microalgas em lagoas com dados meteorológicos incertos, através de um otimizador não-linear de programação quadrática sequencial com resultados satisfatórios que conseguiram manter o sistema em níveis seguros de operação com uma alta produtividade; Malek et al., 2015 em seu estudo de otimização da produção de microalgas em lagoas alcançou uma diminuição de 15% dos custos de produção; Albarello et al., 2019 em seu estudo de otimização do crescimento de microalgas em biorreator batelada, aplicado sobre um modelo matemático, projetou um aumento de até 46% da produtividade seguindo a política de diluição otimizada.

5 Conclusões e Trabalhos Futuros

Este trabalho avaliou o desempenho do método *Actor Critic* na otimização da produção de lipídeos aplicado a um modelo de um biorreator de microalgas comparado a otimizadores dinâmicos. As conclusões desta avaliação são que, pela metodologia aplicada, o método utilizado não é uma alternativa viável aos otimizadores *Cobyla* e *Direct,* uma vez que além de um tempo maior, cerca de 100% e 40% de tempo a mais que os métodos *Direct* e *Cobyla* respectivamente, também apresentou valores finais de produção de lipídeos abaixo dos métodos convencionais de otimização aos quais foi comparado, sendo um pouco maior que a metade dos valores alcançado pelos dois métodos de otimização dinâmica.

No entanto, deve-se entender que, apesar da inviabilidade do método aplicado, novos trabalhos podem trazer novas abordagens na aplicação do aprendizado por reforço aplicado à produção de microalgas, uma vez que este estudo avaliou apenas um método aplicado a um único modelo. Desta forma novos trabalhos podem ser realizados no futuro quanto a aplicação de outros métodos ao modelo utilizado neste estudo, na variação de outros parâmetros do método assim como testes considerando outras faixas de valores para os parâmetros considerados e na consideração dos custos de produção como penalidade na função de recompensa.

REFERÊNCIAS

ABIDEEN, A. Z.; PANDIYAN, V.; SUNDRAM, K.; et al. Digital Twin Integrated Reinforced Learning in Supply Chain and Logistics. **Logistics 2021, Vol. 5, Page 84**, v. 5, n. 4, p. 84, 26 nov. 2021.

ALBARELLO, A.; SIMIONATO, D.; MOROSINOTTO, T.; et al. Model-Based Optimization of Microalgae Growth in a Batch Plant. 2019.

BRENNAN, L.; OWENDE, P. Biofuels from microalgae—A review of technologies for production, processing, and extractions of biofuels and co-products. **Renewable and Sustainable Energy Reviews**, v. 14, n. 2, p. 557–577, 1 fev. 2010.

CHISTI, Y. Biodiesel from microalgae. **Biotechnology Advances**, v. 25, n. 3, p. 294–306, maio 2007.

DE-LUCA, R.; TRABUIO, M.; BAROLO, M.; et al. Microalgae growth optimization in open ponds with uncertain weather data. **Computers & Chemical Engineering**, v. 117, p. 410–419, 2 set. 2018.

DROOP, M. R. Vitamin B_{12} and Marine Ecology. IV. The Kinetics of Uptake, Growth and Inhibition in *Monochrysis Lutheri*. **Journal of the Marine Biological Association of the United Kingdom**, v. 48, n. 3, p. 689–733, 11 out. 1968.

GRAESSER, L. G.; KENG, W. L. **Foundations of Deep Reinforcement Learning: Theory and Practice in Python**. 1st. ed. [s.l.] Addison-Wesley Professional, 2019.

GRONDMAN, I.; BUSONIU, L.; LOPES, G. A. D.; et al. **A survey of actor-critic reinforcement learning: Standard and natural policy gradientsIEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews**, 2012.

H. TAMIYA; E. HASE; K. SHIBATA; et al. Kinetics of growth of Chlorella, with special reference to its dependence on quantity of available light and on temperature. **Algal culture from laboratory to pilot plant**, p. 204–234, 1953.

HE, L.; SUBRAMANIAN, V. R.; TANG, Y. J. Experimental analysis and model-based optimization of microalgae growth in photo-bioreactors using flue gas. **Biomass and Bioenergy**, v. 41, p. 131–138, 1 jun. 2012.

HOSSAIN, S. M. Z.; SULTANA, N.; RAZZAK, S. A.; et al. Modeling and multi-objective optimization of microalgae biomass production and CO2 biofixation using hybrid intelligence approaches. **Renewable and Sustainable Energy Reviews**, v. 157, p. 112016, 1 abr. 2022.

HUANG, X.; ZHANG, D.; ZHANG, X. S. Energy management of intelligent building based on deep reinforced learning. **Alexandria Engineering Journal**, v. 60, n. 1, p. 1509–1517, 1 fev. 2021.

KOTSIANTIS, S. B. Supervised machine learning: a review of classification techniques. **Informatica (Ljubljana)**, v. 31, n. 3, p. 249-, 1 out. 2007.

KUCYK, D. **Reinforcement Learning aplicado para otimização da produção de poços de elevação de petróleo**. Porto Alegre: [s.n.].

KUNIKANE, S.; KANEKO, M. UNDER A WIDE RANGE OF NITROGEN/PHOSPHORUS RATIO-II KINETIC MODEL. **Water Res**, v. 18, n. I0, p. 1313–1326, 1984.
LEE, E.; JALALIZADEH, M.; ZHANG, Q. Growth kinetic models for microalgae cultivation: A review. **Algal Research**, v. 12, p. 497–512, 1 nov. 2015.
LESOT, M. J. Kernel-based outlier preserving clustering with representativity coefficients. **Modern Information Processing**, p. 183–194, 1 jan. 2006.
MALEK, A.; ZULLO, L. C.; DAOUTIDIS, P. Modeling and Dynamic Optimization of Microalgae Cultivation in Outdoor Open Ponds. 2015.
MALEK, A.; ZULLO, L. C.; DAOUTIDIS, P. Modeling and Dynamic Optimization of Microalgae Cultivation in Outdoor Open Ponds. **Industrial and Engineering Chemistry Research**, v. 55, n. 12, p. 3327–3337, 30 mar. 2016.
MATA, T. M.; MARTINS, A. A.; CAETANO, N. S. Microalgae for biodiesel production and other applications: A review. **Renewable and Sustainable Energy Reviews**, v. 14, n. 1, p. 217–232, 1 jan. 2010.
MNIH, V.; PUIGDOMÈNECH BADIA, A.; MIRZA, M.; et al. Asynchronous Methods for Deep Reinforcement Learning. 2016.
MONOD, J. THE GROWTH OF BACTERIAL CULTURES. **Annual Review of Microbiology**, v. 3, n. 1, p. 371–394, 1949.
OGBONNA, J. C.; YADA, H.; TANAKA, H. **Kinetic Study on Light-Limited Batch Cultivation Photosynthetic Cells ofJOURNAL OF FERMENTATION AND BIOENGINEERING**. [s.l: s.n.].
PETSAGKOURAKIS, P.; SANDOVAL, I. O.; BRADFORD, E.; et al. Reinforcement learning for batch bioprocess optimization. **Computers & Chemical Engineering**, v. 133, p. 106649, 2 fev. 2020.
POWELL, B. K. M.; MACHALEK, D.; QUAH, T. Real-time optimization using reinforcement learning. **Computers & Chemical Engineering**, v. 143, p. 107077, 5 dez. 2020.
S. SUTTON, R.; G. BARTO, A. **Reinforcement Learning: An Introduction**. 2nd Edition ed. Cambridge: Bradford Book, 2018.
SAIKIA, L. C.; MISHRA, S.; SINHA, N.; et al. Automatic generation control of a multi area hydrothermal system using reinforced learning neural network controller. **International Journal of Electrical Power & Energy Systems**, v. 33, n. 4, p. 1101–1108, 1 maio 2011.
SIMONINI, T.; SANSEVIERO, O. **The Hugging Face Deep Reinforcement Learning Class**. Disponível em: <https://huggingface.co/blog/deep-rl-a2c>. Acesso em: 29 ago. 2022.
STEELE, J. H. **ENVIRONMENTAL CONTROL OF PHOTOSYNTHESIS IN THE SEA**. [s.l: s.n.].
SZARSKI, M.; CHAUHAN, S. Composite temperature profile and tooling optimization via Deep Reinforcement Learning. **Composites Part A: Applied Science and Manufacturing**, v. 142, p. 106235, 1 mar. 2021.
VIRTANEN, P.; GOMMERS, R.; OLIPHANT, T. E.; et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. **Nature Methods**, v. 17, n. 3, p. 261–272, 1 mar. 2020.

YOO, S. J.; JEONG, D. H.; KIM, J. H.; et al. Optimization of microalgal photobioreactor system using model predictive control with experimental validation. **Bioprocess and biosystems engineering**, v. 39, n. 8, p. 1235–1246, 1 ago. 2016.

APÊNDICE A – *ADVANTAGE ACTOR CRITIC*

As Figuras A.1 a A.12 mostram o código implementado em python para a aplicação do método *Advantage Actor Critic* na otimização da produção de lipídeos de microalgas.

```
from scipy.integrate import odeint
from scipy.optimize import fsolve
import matplotlib.pyplot as plt
from gym.spaces import Box,MultiDiscrete
import numpy as np
import time
import gym
from gym import Env
import os
from stable_baselines3 import A2C
from stable_baselines3.common.evaluation import evaluate_policy
from stable_baselines3.common.callbacks import EvalCallback
```

Figura A.1: Definição das bibliotecas utilizadas.

```
um=0.0418 #max growth rate (L/h)
q0=0.0196 #min nitrogen quota (g/g)
qm=0.2109 #max nitrogen quota (g/g)
l0=0.0006 #min lipid quota (g/g)
lm=0.6995 #max lipid quota (g/g)
Ks2=0.1002 #half saturation constant of glucose (g/L)
Ks1=0.5793 #half saturation constant for glycine (g/L)
Kl=66.5337 #half saturation constant of light (umol/m²s)
Kn=12.5596 #half saturation constant of lipid (g/L)
pm=0.1197 #max uptake rate (1/h)
nm=0.0762 #max lipid production rate (1/h)
Yxs=0.9597 #glucose/biomass yield coef (g/g)
Yls=0.1908 #glucose/lipid yield coef (g/g)

S1i=10. #glycine feed conc (g/L)
S2i=200. #glucose feed conc (g/L)
Vr=3. #volume do reator (L)
V=2

inicial=np.array([0.0016,0.0018,500])
```

Figura A.2: Definição dos parâmetros do modelo matemático utilizado.

```
def D(V):
    return (f1+f2)/V #efeito da diluição, talvez deva ser adicionada a vazão de saída (f1+f2-f0)/V

def X(N,L,x):
    return (N+L+x) #total biomass conc

def l(L):
    return L/X(N,L,x) #lipid quota

def q(N):
    return N/X(N,L,x) #nitrogen quota

def u(S2):
    return um*(1-q0/q(N))*(1-l0/l(L))*(S2/(Ks2+S2))*(I/(Kl+I)) #growth rate

def p(S1):
    return pm*(S1/(Ks1+S1))*((qm-q(N))/(qm-q0)) #nitrogen uptake rate

def n(S2):
    return nm*(S2/(Kn+S2))*(1-N/X(N,L,x))*((lm-l(L))/lm) #lipid production rate
```

Figura A.3: Definição das variáveis não manipuladas.

```
def ODEs(Var,T,f1,f2,I):

    ODEs=np.zeros(5)

    x=Var[0]
    S1=Var[1]
    S2=Var[2]
    N=Var[3]
    L=Var[4]

    f0=f1+f2

    dx=u(S2)*x-D(V)*x
    dS1=-p(S1)*x+(S1i*f1)/V-S1*D(V)
    dS2=-1/Yxs*u(S2)*x-1/Yls*n(S2)*x+(S2i*f2)/V-S2*D(V)
    dN=p(S1)*x-u(S2)*N-N*D(V)
    dL=n(S2)*x-u(S2)*L-L*D(V)


    ODEs[0]=dx
    ODEs[1]=dS1
    ODEs[2]=dS2
    ODEs[3]=dN
    ODEs[4]=dL

    return ODEs
```

Figura A.4: Definição das equações de estado do modelo utilizado.

```
f1=inicial[0]
f2=inicial[1]
I=inicial[2]
lip=0

Var=np.zeros(5)
Var[0]=0.1
Var[1]=10./2
Var[2]=200./2
Var[3]=0.02
Var[4]=0.001

x=Var[0]
S1=Var[1]
S2=Var[2]
N=Var[3]
L=Var[4]
```

Figura A.5: Definição dos valores para a inicialização do método.

```
class MicAlg(Env):
    metadata = {'render.modes': ['human']}
    def __init__(self):
        global f1
        global f2
        global f0
        global I
        global Var
        global t
        global lip
        lip=0
        self.action_space = MultiDiscrete([20,20,20])
        self.observation_space = Box(low=np.array([0,0,0,0,0,0,0,10,0]),
                                     high=np.array([20,20,200,20,20,0.01,0.01,750,np.inf]),
                                     shape=(9,))
        self.state = (Var[0],Var[1],Var[2],Var[3],Var[4],f1,f2,I,lip)
        t=0

    def step(self,action,):
        global f1
        global f2
        global f0
        global I
        global Var
        global t
        global steptfinal
        global steptime
        global lip
        perc=np.array([0.01/100, 0.01/100, 740/100])
        red = np.array([10,10,10])
        if t>0:
            actf = (action-red)*perc
        else:
            actf = [0,0,0]
        steptfinal=30
        steptime=100
```

Figura A.6: Criação do ambiente *Gym* e definição do espaço de ações.

```
f1 += actf[0]
f2 += actf[1]
I += actf[2]

if f1<0.:
    f1=0.
elif f1>0.01:
    f1=0.01

if f2<0.:
    f2=0.
elif f2>0.01:
    f2=0.01

if I<10:
    I=10
elif I>750:
    I=750

f0 = f1+f2
h=steptime
stept=np.linspace(0,steptime/10,steptime+1)
stepres = odeint(ODEs,Var,stept,args=(f1,f2,I))

Var[0]=stepres[h,0]
Var[1]=stepres[h,1]
Var[2]=stepres[h,2]
Var[3]=stepres[h,3]
Var[4]=stepres[h,4]

if Var[3]<0:
    Var[3]=0
if Var[1]>10:
    Var[1]=10
if Var[2]>200:
    Var[2]=200
```

Figura A.7: Resolução das equações diferenciais após tomada da decisão.

```
rewardx=np.zeros(h)
rewardx=stepres[0:h,4]*f0*0.1 #Concentração de lipídeos no interior celular
reward=np.sum(rewardx)

t+=1
self.state=(Var[0],Var[1],Var[2],Var[3],Var[4],f1,f2,I)

if t>=steptfinal:
    done=True
else:
    done=False

info={}

return self.state, reward, done, info
```

Figura A.8: Cálculo da recompensa e retorno dos resultados do passo.

```
def render(self):
    pass

def reset(self):
    global f1
    global f2
    global f0
    global I
    global Var
    global t
    global lip
    lip=0
    f1=inicial[0]
    f2=inicial[1]
    I=inicial[2]
    t=0
    Var[0]=0.1
    Var[1]=10./2
    Var[2]=200./2
    Var[3]=0.02
    Var[4]=0.001

    self.state=(Var[0],Var[1],Var[2],Var[3],Var[4],f1,f2,I,lip)

    return self.state
```

Figura A.9: Finalização da criação do ambiente *Gym*.

```
env = MicAlg()
save_path = os.path.join('Saved Models 2','A2C 100000 0.005lr 4')
eval_callback = EvalCallback(env,
                             eval_freq = 10000,
                             best_model_save_path = save_path,
                             log_path = save_path,
                             verbose = 1)
log_path = os.path.join('Logs')
model = A2C('MlpPolicy', env, verbose=1, tensorboard_log="./Logs 2/",
            learning_rate=0.005, seed=13)

start_time = time.time()

best_model_path = os.path.join(save_path,"best_model")
model.learn(total_timesteps=100000, callback = eval_callback,
            tb_log_name="A2C 100000 0.005lr 4")

tempo_op = (time.time() - start_time)

del model
```

Figura A.10: Configuração do aprendizado na biblioteca *stable baselines 3*.

```
best_model_path = os.path.join(save_path,"best_model")
model = A2C.load(best_model_path, env=env)
evaluate_policy(model, env, n_eval_episodes = 10, render=False)

Resultados = np.zeros((int(steptfinal),9))
episodes = 1
score_med=0
for episodes in range(11):
    obs=env.reset()
    done=False
    score=0

    time = 0
    X_hist = np.zeros(30)
    while (not done):
        action, _ = model.predict(obs)
        obs,reward,done,info = env.step(action)
        Resultados[time,:] = obs
        X_hist[time] = X(Resultados[time,3],Resultados[time,4],Resultados[time,0])
        time += 1
        score+=reward
    print('Episode:{} Score:{}'.format(episodes,score))
    score_med+=score
print('score médio: ',score_med/11)
print('tempo de execução: ', tempo_op)
time=np.linspace(0,(time)*10,time+1)
```

Figura A.11: Teste do método e coleta dos resultados obtidos.

```
plt.plot(time,Resultados[:,4])
plt.xlabel('Tempo (h)')
plt.ylabel('Concentração de Lipídeos (g/L)')
plt.grid()
plt.show()

plt.step(time,Resultados[:,5])
plt.xlabel('Tempo (h)')
plt.ylabel('f1 (L/h)')
plt.grid()
plt.show()

plt.step(time,Resultados[:,6])
plt.xlabel('Tempo (h)')
plt.ylabel('f2 (L/h)')
plt.grid()
plt.show()

plt.step(time,Resultados[:,7])
plt.xlabel('Tempo (h)')
plt.ylabel('I (µmol/m²s)')
plt.grid()
plt.show()
```

Figura A.12: Geração dos gráficos do método *Advantage Actor Critic*.

APÊNDICE B - *COBYLA*

As Figuras B.1 a B.11 mostram o código implementado em python para a aplicação do método de otimização *Cobyla* à produção de lipídeos de microalgas.

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from scipy.optimize import minimize, fsolve
import random
import time
```

Figura B.1: Definição das bibliotecas utilizadas.

```
um=0.0418 #max growth rate (L/h)
q0=0.0196 #min nitrogen quota (g/g)
qm=0.2109 #max nitrogen quota (g/g)
l0=0.0006 #min lipid quota (g/g)
lm=0.6995 #max lipid quota (g/g)
Ks2=0.1002 #half saturation constant of glucose (g/L)
Ks1=0.5793 #half saturation constant for glycine (g/L)
Kl=66.5337 #half saturation constant of light (umol/m²s)
Kn=12.5596 #half saturation constant of lipid (g/L)
pm=0.1197 #max uptake rate (1/h)
nm=0.0762 #max lipid production rate (1/h)
Yxs=0.9597 #glucose/biomass yield coef (g/g)
Yls=0.1908 #glucose/lipid yield coef (g/g)

S1i=10. #glycine feed conc (g/L)
S2i=200. #glucose feed conc (g/L)
Vr=3. #volume do reator (L)
V=2

inicial=np.array([0.0016,0.0018,500])
```

Figura B.2: Definição dos parâmetros do modelo matemático utilizado.

```
def D(V):
    return (f1+f2)/V #efeito da diluição, talvez deva ser adicionada a vazão de saída (f1+f2-f0)/V

def X(N,L,x):
    return (N+L+x) #total biomass conc

def l(L):
    return L/X(N,L,x) #lipid quota

def q(N):
    return N/X(N,L,x) #nitrogen quota

def u(S2):
    return um*(1-q0/q(N))*(1-l0/l(L))*(S2/(Ks2+S2))*(I/(Kl+I)) #growth rate

def p(S1):
    return pm*(S1/(Ks1+S1))*((qm-q(N))/(qm-q0)) #nitrogen uptake rate

def n(S2):
    return nm*(S2/(Kn+S2))*(1-N/X(N,L,x))*((lm-l(L))/lm) #lipid production rate
```

Figura B.3: Definição das variáveis não manipuladas.

```
def ODEs(Var,T,f1,f2,I):

    ODEs=np.zeros(5)

    x=Var[0]
    S1=Var[1]
    S2=Var[2]
    N=Var[3]
    L=Var[4]

    f0=f1+f2

    dx=u(S2)*x-D(V)*x
    dS1=-p(S1)*x+(S1i*f1)/V-S1*D(V)
    dS2=-1/Yxs*u(S2)*x-1/Yls*n(S2)*x+(S2i*f2)/V-S2*D(V)
    dN=p(S1)*x-u(S2)*N-N*D(V)
    dL=n(S2)*x-u(S2)*L-L*D(V)


    ODEs[0]=dx
    ODEs[1]=dS1
    ODEs[2]=dS2
    ODEs[3]=dN
    ODEs[4]=dL

    return ODEs
```

Figura B.4: Definição das equações de estado do modelo utilizado.

```
Arg=np.zeros(93)
Arg_hist=np.zeros(93)
f1=inicial[0]
f2=inicial[1]
I=inicial[2]
Arg[0]=f1
Arg[1]=f2
Arg[2]=I
lip=0

Var=np.zeros((3001,5))
Var[0,0]=0.1
Var[0,1]=40.
Var[0,2]=200.
Var[0,3]=0.02
Var[0,4]=0.001


x=Var[0,0]
S1=Var[0,1]
S2=Var[0,2]
N=Var[0,3]
L=Var[0,4]

steptfinal=30
stept=100
i=3
random.seed(a=1023453210)
```

Figura B.5: Definição dos valores de inicialização do método *Cobyla*.

```
def func_otm(Arg):
    global Var
    Var_func=np.zeros(5)
    Var_func=Var[0,:]
    i=1
    j=3
    reward=np.zeros(steptfinal)
    steplin=np.linspace(0,stept/10,stept+1)
    Var[0,:] = odeint(ODEs,Var_func,steplin,args=(Arg[0],Arg[1],Arg[2]))[0,:]
    while i in range(steptfinal+1):
        if Arg[j]>20:
            Arg[j]=20
        elif Arg[j]<0.:
            Arg[j]=0.
        if Arg[j+1]>20:
            Arg[j+1]=20
        elif Arg[j+1]<0.:
            Arg[j+1]=0.
        if Arg[j+2]>20:
            Arg[j+2]=20
        elif Arg[j+2]<0.:
            Arg[j+2]=0.
        Arg[j]= Arg[j-3]+(Arg[j]-10)*0.01/100
        Arg[j+1]=Arg[j-2]+(Arg[j+1]-10)*0.01/100
        Arg[j+2]=Arg[j-1]+(Arg[j+2]-10)*740/100
        if Arg[j]>0.01:
            Arg[j]=0.01
        elif Arg[j]<0.:
            Arg[j]=0.
        if Arg[j+1]>0.01:
            Arg[j+1]=0.01
        elif Arg[j+1]<0.:
            Arg[j+1]=0.
        if Arg[j+2]>750:
            Arg[j+2]=750
        elif Arg[j+2]<10.:
            Arg[j+2]=10.
        Var[(i-1)*stept:i*stept+1,:] = odeint(ODEs,Var_func,steplin,
                                              args=(Arg[j],Arg[j+1],Arg[j+2]))
        k=0
        while k in range(stept+1):
            if Var[k+10*(i-1),3]<0:
                Var[k+10*(i-1),3]=0
            if Var[k+10*(i-1),1]>10:
                Var[k+10*(i-1),1]=10
            if Var[k+10*(i-1),2]>200:
                Var[k+10*(i-1),2]=200
            k+=1
        Var_func = Var[i*stept,:]
        f0=Arg[j]+Arg[j+1]
        reward[i-1] = np.sum(Var[(i-1)*stept:i*stept,4]*f0*0.1)
        i+=1
        j+=3
    reward_total = -np.sum(reward)
    return reward_total
```

Figura B.6: Definição da função objetivo para o método *Cobyla*.

```
def constraints():
    boundsf1 = (0,0.01)
    boundsf2 = (0,0.01)
    boundsI = (10,750)
    bound = []
    i = 0
    while i < steptfinal+1:
        bound.append(boundsf1)
        bound.append(boundsf2)
        bound.append(boundsI)
        i+=1

    cons = []
    j=0
    while j in range(len(bound)):
        lower1, upper1 = bound[j]
        lower2, upper2 = bound[j+1]
        lower3, upper3 = bound[j+2]
        low1 = {'type': 'ineq', 'fun': lambda Arg, lb=lower1, i=j: Arg[i] - lb}
        high1 = {'type': 'ineq', 'fun': lambda Arg, ub=upper1, i=j: ub - Arg[i]}
        low2 = {'type': 'ineq', 'fun': lambda Arg, lb=lower2, i=j+1: Arg[i] - lb}
        high2 = {'type': 'ineq', 'fun': lambda Arg, ub=upper2, i=j+1: ub - Arg[i]}
        low3 = {'type': 'ineq', 'fun': lambda Arg, lb=lower3, i=j+2: Arg[i] - lb}
        high3 = {'type': 'ineq', 'fun': lambda Arg, ub=upper3, i=j+2: ub - Arg[i]}
        cons.append(low1)
        cons.append(high1)
        cons.append(low2)
        cons.append(high2)
        cons.append(low3)
        cons.append(high3)
        j+=3
    return cons
```

Figura B.7: Definição das restrições para o método *Cobyla*.

```
cons = constraints()

start_time = time.time()
sol = minimize(func_otm, x0=Arg , method='COBYLA', options={'maxiter':100000}, bounds=cons)
time_op = (time.time() - start_time)

sol_res = sol
```

Figura B.8: Resolução do método *Cobyla*.

```
def func_test(Arg):
    global Arg_hist
    Arg_hist[0]=Arg[0]
    Arg_hist[1]=Arg[1]
    Arg_hist[2]=Arg[2]
    Var_func=np.zeros(5)
    Var_func=Var[0,:]
    i=1
    j=3
    reward=np.zeros(steptfinal)
    rates=np.zeros((steptfinal+1,7))
    steplin=np.linspace(0,stept/10,stept+1)
    Var[0,:] = odeint(ODEs,Var_func,steplin,args=(Arg[0],Arg[1],Arg[2]))[0,:]
    while i in range(steptfinal+1):
        Arg[j]= Arg[j-3]+(Arg[j]-10)*0.01/100
        Arg[j+1]=Arg[j-2]+(Arg[j+1]-10)*0.01/100
        Arg[j+2]=Arg[j-1]+(Arg[j+2]-10)*740/100
        Arg_hist[j]=Arg[j]
        Arg_hist[j+1]=Arg[j+1]
        Arg_hist[j+2]=Arg[j+2]
        Var[(i-1)*stept:i*stept+1,:] = odeint(ODEs,Var_func,steplin,
                                              args=(Arg[j],Arg[j+1],Arg[j+2]))
        Var_func = Var[i*stept,:]
        f0=Arg[j]+Arg[j+1]
        k=0
        while k in range(stept+1):
            if Var[k+10*(i-1),3]<0:
                Var[k+10*(i-1),3]=0
            if Var[k+10*(i-1),1]>10:
                Var[k+10*(i-1),1]=10
            if Var[k+10*(i-1),2]>200:
                Var[k+10*(i-1),2]=200
            k+=1
        k=0
        while k in range(steptfinal+1):
            rates[i,0] = u(Var_func[2]) #u
            rates[i,1] = p(Var_func[1]) #p
            rates[i,2] = n(Var_func[2]) #n
            rates[i,3] = D(V) #D
            rates[i,4] = X(Var_func[0],Var_func[3],Var_func[4]) #X
            rates[i,5] = l(Var_func[4]) #l
            rates[i,6] = q(Var_func[3]) #q
            k+=1
        reward[i-1] = np.sum(Var[(i-1)*stept:i*stept,4]*f0*0.1)
        i+=1
        j+=3
    reward_total = -np.sum(reward)
    return Var, rates, reward_total

Resultados, rates, recompensa = func_test(sol_res.x)
```

Figura B.9: Definição da função utilizada para testar os valores ótimos das variáveis manipuladas do método *Cobyla*.

```
i=0
j=0
f1_hist=np.zeros(int(len(sol.x)/3))
f2_hist=np.zeros(int(len(sol.x)/3))
I_hist=np.zeros(int(len(sol.x)/3))
while i in range(int(len(sol.x))):
    f1_hist[j]=Arg_hist[i]
    f2_hist[j]=Arg_hist[i+1]
    I_hist[j]=Arg_hist[i+2]
    i+=3
    j+=1
```

Figura B.10: Separação dos resultados do método *Cobyla* para a geração dos gráficos.

```
time = np.linspace(0,300,steptfinal+1)
timet = np.linspace(0,300,stept*30+1)

plt.plot(timet,Resultados[:,4])
plt.xlabel('Tempo (h)')
plt.ylabel('Concentração de Lipídeos (g/L)')
plt.grid()
plt.show()

plt.step(time,f1_hist)
plt.xlabel('Tempo (h)')
plt.ylabel('f1 (L/h)')
plt.grid()
plt.show()

plt.step(time,f2_hist)
plt.xlabel('Tempo (h)')
plt.ylabel('f2 (L/h)')
plt.grid()
plt.show()

plt.step(time,I_hist)
plt.xlabel('Tempo (h)')
plt.ylabel('I (µmol/m²s)')
plt.grid()
plt.show()
```

Figura B.11: Geração dos gráficos dos resultados do método *Cobyla*.

APÊNDICE C - *DIRECT*

As Figuras C.1 a C.11 mostram o código implementado em python para a aplicação do método de otimização *Direct* à produção de lipídeos de microalgas.

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from scipy.optimize import direct, fsolve
import random
```

Figura C.1: Definição das bibliotecas utilizadas.

```
um=0.0418 #max growth rate (L/h)
q0=0.0196 #min nitrogen quota (g/g)
qm=0.2109 #max nitrogen quota (g/g)
l0=0.0006 #min lipid quota (g/g)
lm=0.6995 #max lipid quota (g/g)
Ks2=0.1002 #half saturation constant of glucose (g/L)
Ks1=0.5793 #half saturation constant for glycine (g/L)
Kl=66.5337 #half saturation constant of light (umol/m²s)
Kn=12.5596 #half saturation constant of lipid (g/L)
pm=0.1197 #max uptake rate (1/h)
nm=0.0762 #max lipid production rate (1/h)
Yxs=0.9597 #glucose/biomass yield coef (g/g)
Yls=0.1908 #glucose/lipid yield coef (g/g)

S1i=10. #glycine feed conc (g/L)
S2i=200. #glucose feed conc (g/L)
Vr=3. #volume do reator (L)
V=2

inicial=np.array([0.0016,0.0018,500])
```

Figura C.2: Definição dos parâmetros do modelo matemático utilizado.

```
def D(V):
    return (f1+f2)/V #efeito da diluição, talvez deva ser adicionada a vazão de saída (f1+f2-f0)/V

def X(N,L,x):
    return (N+L+x) #total biomass conc

def l(L):
    return L/X(N,L,x) #lipid quota

def q(N):
    return N/X(N,L,x) #nitrogen quota

def u(S2):
    return um*(1-q0/q(N))*(1-l0/l(L))*(S2/(Ks2+S2))*(I/(Kl+I)) #growth rate

def p(S1):
    return pm*(S1/(Ks1+S1))*((qm-q(N))/(qm-q0)) #nitrogen uptake rate

def n(S2):
    return nm*(S2/(Kn+S2))*(1-N/X(N,L,x))*((lm-l(L))/lm) #lipid production rate
```

Figura C.3: Definição das variáveis não manipuladas.

```
def ODEs(Var,T,f1,f2,I):

    ODEs=np.zeros(5)

    x=Var[0]
    S1=Var[1]
    S2=Var[2]
    N=Var[3]
    L=Var[4]

    f0=f1+f2

    dx=u(S2)*x-D(V)*x
    dS1=-p(S1)*x+(S1i*f1)/V-S1*D(V)
    dS2=-1/Yxs*u(S2)*x-1/Yls*n(S2)*x+(S2i*f2)/V-S2*D(V)
    dN=p(S1)*x-u(S2)*N-N*D(V)
    dL=n(S2)*x-u(S2)*L-L*D(V)


    ODEs[0]=dx
    ODEs[1]=dS1
    ODEs[2]=dS2
    ODEs[3]=dN
    ODEs[4]=dL

    return ODEs
```

Figura C.4: Definição das equações de estado do modelo utilizado.

```
Arg=np.zeros(93)
Arg_hist=np.zeros(93)
f1=inicial[0]
f2=inicial[1]
I=inicial[2]
Arg[0]=f1
Arg[1]=f2
Arg[2]=I

Var=np.zeros((301,5))
Var[0,0]=0.1
Var[0,1]=40.
Var[0,2]=200.
Var[0,3]=0.02
Var[0,4]=0.001


x=Var[0,0]
S1=Var[0,1]
S2=Var[0,2]
N=Var[0,3]
L=Var[0,4]

steptfinal=30
stept=10
i=0
random.seed(a=1023453210)
```

Figura C.5: Definição dos valores de inicialização do método *Direct*.

```
def func_otm(Arg):
    global Var
    Var_func=np.zeros(5)
    Var_func=Var[0,:]
    i=1
    j=3
    reward=np.zeros(steptfinal)
    steplin=np.linspace(0,stept,stept+1)
    Var[0,:] = odeint(ODEs,Var_func,steplin,args=(Arg[0],Arg[1],Arg[2]))[0,:]
    while i in range(steptfinal+1):
        Arg[j]= Arg[j-3]+(Arg[j]-10)*0.01/100
        Arg[j+1]=Arg[j-2]+(Arg[j+1]-10)*0.01/100
        Arg[j+2]=Arg[j-1]+(Arg[j+2]-10)*740/100
        if Arg[j]>0.01:
            Arg[j]=0.01
        elif Arg[j]<0.:
            Arg[j]=0.
        if Arg[j+1]>0.01:
            Arg[j+1]=0.01
        elif Arg[j+1]<0.:
            Arg[j+1]=0.
        if Arg[j+2]>750:
            Arg[j+2]=750
        elif Arg[j+2]<10.:
            Arg[j+2]=10.
        Var[(i-1)*stept:i*stept+1,:] = odeint(ODEs,Var_func,steplin,args=(Arg[j],Arg[j+1],Arg[j+2]))
        k=0
        while k in range(stept+1):
            if Var[k+10*(i-1),3]<0:
                Var[k+10*(i-1),3]=0
            if Var[k+10*(i-1),1]>10:
                Var[k+10*(i-1),1]=10
            if Var[k+10*(i-1),2]>200:
                Var[k+10*(i-1),2]=200
            k+=1
        Var_func = Var[i*stept,:]
        reward[i-1] = np.sum(Var[(i-1)*stept:i*stept,4])
        i+=1
        j+=3
    reward_total = -np.sum(reward)
    return reward_total
```

Figura C.6: Definição da função objetivo para o método *Direct*.

```
def constraints():
    boundsf1 = (0,20)
    boundsf2 = (0,20)
    boundsI = (0,20)
    bound = [(0,0.01),(0,0.01),(10,750)]
    i = 1
    while i < steptfinal+1:
        bound.append(boundsf1)
        bound.append(boundsf2)
        bound.append(boundsI)
        i+=1

    return bound
```

Figura C.7: Definição das restrições para o método *Direct*.

```
cons = constraints()

sol = direct(func_otm, bounds=cons, maxiter=1000)
sol_res = sol
```

Figura C.8: Resolução do método *Direct*.

```
def func_test(Arg):
    global Arg_hist
    global lip

    Arg_hist[0]=Arg[0]
    Arg_hist[1]=Arg[1]
    Arg_hist[2]=Arg[2]
    Var_func=np.zeros(5)
    Var_func=Var[0,:]
    i=1
    j=3
    reward=np.zeros(steptfinal)
    rates=np.zeros((steptfinal+1,7))
    steplin=np.linspace(0,stept/10,stept+1)
    Var[0,:] = odeint(ODEs,Var_func,steplin,args=(Arg[0],Arg[1],Arg[2]))[0,:]
    while i in range(steptfinal+1):
        Arg[j]= Arg[j-3]+(Arg[j]-10)*0.01/100
        Arg[j+1]=Arg[j-2]+(Arg[j+1]-10)*0.01/100
        Arg[j+2]=Arg[j-1]+(Arg[j+2]-10)*740/100
        Arg_hist[j]=Arg[j]
        Arg_hist[j+1]=Arg[j+1]
        Arg_hist[j+2]=Arg[j+2]
        Var[(i-1)*stept:i*stept+1,:] = odeint(ODEs,Var_func,steplin,
                                              args=(Arg[j],Arg[j+1],Arg[j+2]))
        Var_func = Var[i*stept,:]
        f0=Arg[j]+Arg[j+1]
        k=0
        while k in range(stept+1):
            if Var[k+10*(i-1),3]<0:
                Var[k+10*(i-1),3]=0
            if Var[k+10*(i-1),1]>10:
                Var[k+10*(i-1),1]=10
            if Var[k+10*(i-1),2]>200:
                Var[k+10*(i-1),2]=200
            k+=1
        k=0
        while k in range(steptfinal+1):
            rates[i,0] = u(Var_func[2]) #u
            rates[i,1] = p(Var_func[1]) #p
            rates[i,2] = n(Var_func[2]) #n
            rates[i,3] = D(V) #D
            rates[i,4] = X(Var_func[0],Var_func[3],Var_func[4]) #X
            rates[i,5] = l(Var_func[4]) #l
            rates[i,6] = q(Var_func[3]) #q
            k+=1
        reward[i-1] = np.sum(Var[(i-1)*stept:i*stept,4]*f0*0.1)
        # lip+=np.sum(Var[(i-1)*stept:i*stept,4]*f0*0.1)
        i+=1
        j+=3
    reward_total = -np.sum(reward)
    return Var, rates, reward_total

Resultados, rates, recompensa = func_test(sol.x)
```

Figura C.9: Definição da função utilizada para testar os valores ótimos das variáveis manipuladas do método *Direct*.

```
i=0
j=0
f1_hist=np.zeros(int(len(sol.x)/3))
f2_hist=np.zeros(int(len(sol.x)/3))
I_hist=np.zeros(int(len(sol.x)/3))
while i in range(int(len(sol.x))):
    f1_hist[j]=Arg_hist[i]
    f2_hist[j]=Arg_hist[i+1]
    I_hist[j]=Arg_hist[i+2]
    i+=3
    j+=1
```

Figura C.10: Separação dos resultados do método *Direct* para a geração dos gráficos.

```
time = np.linspace(0,300,steptfinal+1)
timet = np.linspace(0,300,stept*30+1)

plt.plot(timet,Resultados[:,4])
plt.xlabel('Tempo (h)')
plt.ylabel('Concentração de Lipídeos (g/L)')
plt.grid()
plt.show()
plt.step(time,f1_hist)
plt.xlabel('Tempo (h)')
plt.ylabel('f1 (L/h)')
plt.grid()
plt.show()

plt.step(time,f2_hist)
plt.xlabel('Tempo (h)')
plt.ylabel('f2 (L/h)')
plt.grid()
plt.show()

plt.step(time,I_hist)
plt.xlabel('Tempo (h)')
plt.ylabel('I (µmol/m²s)')
plt.grid()
plt.show()
```

Figura C.11: Geração dos gráficos dos resultados do método *Direct*.