P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

M

MESTRADO
Engenharia Informática

Otimização do Processo de Receção e Processamento de Uvas em Lagares
Jorge Filipe Santos Pereira

11/2019

P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO
POLITÉCNICO
DO PORTO

M

MESTRADO
Engenharia Informática

Otimização do Processo de Receção e Processamento de Uvas em Lagares
Jorge Filipe Santos Pereira

11/2019

Jorge Filipe Santos Pereira. Otimização do Processo de Receção e Processamento de Uvas em Lagares

Agradecimentos

Primeiramente gostaria de agradecer aos meus orientadores, o professor Davide Carneiro e a professora Eliana Costa e Silva, pela disponibilidade, preocupação, acompanhamento e orientação ao longo da desenvolvimento desta dissertação. Foram essenciais para a conclusão da mesma e não poderia estar mais grato pela ajuda.

De seguida queria agradecer aos meus colegas de curso, que conseguem tornar divertida e produtiva qualquer noitada intensiva de trabalho. Sou grato pelo companheirismo e amizade que construí com cada um ao longo deste nosso percurso académico.

Por fim, não poderia deixar de agradecer aos meus pais e irmã, pela constante preocupação e suporte que deram, não só ao longo do desenvolvimento da dissertação, mas também ao longo de todo o meu percurso académico. Sem vocês eu não estaria aqui hoje.

Um muito obrigado a todos!

Resumo

Durante o período de vindimas os produtores de uvas enfrentam longos peíodos de espera para realizar a descarga das suas uvas nas empresas produtoras de vinhos. Estes tempos de espera têm um impacto negativo na qualidade dos vinhos. O objetivo deste projeto consiste em estudar e utilizar algoritmos de otimização para resolver este problema. Neste sentido, foi desenvolvido um Algoritmo Genético, que através de um conjunto de *inputs* iniciais, encontrar a melhor solução para a otimização do escalonamento dos camiões dos produtores, assim como otimizar o processamento das uvas. Após o Algoritmo Genético obter a solução mais apta para um certo cenário, os colaboradores responsáveis pela receção dos camiões de uvas poderão então visualizar informações sobre a mesma através de dashboards criados no Kibana e que permitem verificar várias informações, tais como: as ações que cada entidade (camiões, tegões e prensas) deve efetuar e quando, a quantidade de uvas de cada entidade ao longo do tempo e tempos médias de espera por tipo de uva. Com os resultados obtidos e com a ajuda das visualizações gráficas dos mesmos, haverá um melhor uso da capacidade de produção disponível, otimização do espaço utilizado durante todo o processo e diminuição do tempo de espera dos produtores, evitando assim, a perda de qualidade dos vinhos.

Palavras-chave: Algoritmos Genéticos, Otimização, Escalonamento, Produtores de vinho

Abstract

During the grape harvest period, producers face long waiting periods to unload their grapes at the winemaking companies. There waiting times have negative impact on wine quality. The aim of this project is to study and use optimization algorithms to solve this problem. In this project Genetic Algorithms were chosen and one was developed. This algorithm is capable of through a set of initial inputs, find the best solution for optimizing the truck scheduling of the producers of these companies, as well as optimizing grape processing. After the Genetic Algorithm gets the most suitable solution for a given scenario, users can then view the information about it through dashboards created in Kibana, that allows them to check various information, such as: the actions that each entity (truck, grain-tank and press) should take and when, quantity of grapes from each entity over time and average waiting times per grape type. With the results obtained by the Genetic Algorithm and with the help of their graphic visualizations, there will be a better use of the available production capacity, optimization of the space used throughout the process and reduction of the waiting time of the producers, thus avoiding the loss of quality of the wines.

Keywords: Genetic Algorithms, Optimization, Scheduling, Wine Producers

Conteúdo

Lista de Tabelas

Lista de Figuras

Lista de Acrónimos

ACO *Ant Colony Optimization*.

AS *Ant System*.

BA *Bat Algotithm*.

CIS *Canterbury Irrigation Scheduler*.

DE Evolução Diferencial.

EA Algoritmos Evolucionérios.

FA *Firefly Algorithm*.

FFT *Fast Fourier Transform*.

GA Algoritmo Genético.

HS *Harmony Search*.

NN Rede Neural.

PSO *Particle Swarm Optimization*.

SA *Simulated Annealing*.

SVM *Support Vector Machine*.

1 Introdução

Ao longo da história a agricultura já passou por várias revoluções e, em todas elas, esteve no estado da arte da tecnologia. É o caso da invenção do arado, há quase cinco mil anos, e da mecanização, no século passado [8]. Portugal é um país com uma longa tradição vinícola, tendo iniciando a sua história com a implantação da vinicultura pelos romanos. Nos últimos anos, com a instalação do Mercado Comum Europeu, o vinho português tem sido cada vez mais procurado pela sua qualidade e visibilidade internacional, tendo conquistado bastante reconhecimento pela qualidade e diversidade dos seus vinhos [9]. A vinicultura Portuguesa demorou a evoluir tecnologicamente, mas nas últimas décadas, como consequência do importante desenvolvimento económico, social e politico do país, a vinicultura portuguesa teve uma grande evolução, particularmente no campo tecnológico. Fato importante é que essa modernização foi realizada sem descartar os aspectos tradicionais positivos, como por exemplo, a utilização de variedades de uvas autóctones e tradicionais. Com ajuda da tecnologia, essas castas, que antes originavam vinhos de qualidade inferior, passaram a dar grandes vinhos, aperfeiçoando as suas características únicas. Diversos enólogos portugueses despontam como artistas no cenário mundial, com vários rótulos conceituados nacionalmente a ganhar prestígio internacional e ocupando um merecido espaço no mercado global [10]. Recentes mudanças na regulamentação impulsionaram novo valor aos Vinhos Regionais, situados entre os vinhos básicos e os classificados, que apresentam uma ótima relação entre custo e benefício, tornando-se cada vez mais competitivos pela associação com a personalidade marcante dos vinhos de Portugal [10].

Com a recente difusão da eletrónica e da automação, mais uma vez a agricultura utilizou a tecnologia de ponta para dar um salto de produtividade. Hoje, a Inteligência Artificial (IA) e a Internet das Coisas (IOT) estão presentes no nosso quotidiano. No que diz respeito à Inteligência Artificial, existem robôs que atendem em *call centers*, ou fazem um diagnóstico médico básico e despertadores que antecipam o toque se houver mais trânsito na nossa rota de ida para o trabalho. No caso da IOT, é possível ver carros que permitem controlar as fechaduras das portas ou ligar o ar condicionado através de uma aplicação e lâmpadas que acendem por comandos de voz. A tecnologia pode ser usada em qualquer área e a agricultura não é exceção. Há algum tempo, através de análise de imagens de drones ou satélites, a IA já nos ajuda com inventário de uma floresta, detecta áreas de ataque de pragas, estima produtividade de uma lavoura e, mais recentemente, começou a entrar nas máquinas, que estão a tornar-se cada vez mais autónomas [11].

Os ganhos de produtividade aumentam dia após dia nas mais inúmeras vertentes, como por exemplo: já se sabe que a manutenção preventiva é mais barata que a manutenção corretiva, mas

agora envolvendo tecnologia, existe a manutenção preditiva. Como exemplo, a partir dela, um certo campo agrícola não será irrigado com uma certa regularidade. Este será irrigado sempre que um sistema considere necessário, sendo este o caso em que IA e IOT poderão ser usados em simultâneo para o efeito. Tendo em conta os dados recolhidos pelos sensores instalados no campo, obtenção de dados por parte de entidades externas (ex. informação metereológica) e a análise destes dados através de processos de machine learning, é possível prever quando é que o campo agrícola irá necessitar de ser irrigado novamente. Ainda existe muito espaço para melhoria de processos atuais, em particular no setor vinícola.

1.1 Descrição do problema

Numa adega, a receção das uvas é um processo em que as uvas colhidas pelos fornecedores, são adquiridas de modo a serem processadas para a posterior produção de vinho. Neste processo existem 4 elementos chave, representados na Figura 1: uvas, camiões (que pertencem aos fornecedores), tegões e prensas (que pertencem à adega). As uvas chegam à adega em camiões dos produtores de vinho, esperando em filas, muitas vezes longas, para efetuar a sua descarga. Quando um camião chega à adega, esta pesa as uvas e mede o seu teor de álcool. Depois, o camião espera no parque até que exista um tegão disponível para que as uvas possam ser descarregadas no mesmo. A principal função do tegão é separar as bagas de uvas das suas hastes e enviar as bagas para as prensas, estando o mesmo encontra-se representado na Figura 2. O processo acaba quando as prensas esmagam as uvas, podendo um exemplo da mesma ser visualizado na Figura 3.

Figura 1: Conceitos fundamentais (da esquerda para a direita): Uvas, camiões, tegões e prensas

O principal objetivo de todo este processo é minimizar o tempo de espera das uvas até que sejam processadas, visto que estas deterioram a sua qualidade e consequentemente, a qualidade do vinho produzido. No entanto, a otimização do processo de receção das uvas, não é um processo simples, tendo em conta o enorme número de variedades e restrições que existem. Alguma restrições são internas da adega, e podem ser, até certo ponto, geridas. No entanto, a maior parte delas são externas e estão fora do controlo da adega. A adega deve gerir o mais eficientemente possível os

seus recursos a todo momento, para que possa reagir a qualquer mudança de variáveis e otimizar todo o processo.

Figura 2: Tegão - Fonte: [1]

Figura 3: Exemplo de uma prensa de uvas [2]

Internamente, as restrições do problema derivam do número de máquinas disponíveis para realizar o processamento das uvas. Isto é, o número de tegões e prensas disponíveis. Quanto mais alto é o número de tegões e prensas, mais fácil será satisfazer as restrições e mais eficientemente as uvas serão processadas. O número de máquinas é normalmente fixo, apesar de poder variar em certas circunstâncias (ex. no caso de avaria de um tegão).

Cada máquina também tem distintas funcionalidades, que podem restringir o processo mais adiante. A capacidade e quantidade de uvas carregadas pelos camiões pode variar significativamente, geralmente num intervalo de 3 a 5 toneladas. Os camiões também podem ser categorizados em dois tipos, basculantes e não basculantes. Ambos geralmente têm uma caixa aberta onde as uvas são carregadas. No entanto, a caixa de um camião basculante é articulada na parte traseira e possui uma ou mais bombas hidráulicas que permitem levantar a frente e descarregar as uvas para dentro do tegão. O camião não basculante, por outro lado, não tem esta habilidade. As uvas são geralmente são carregadas dentro de caixas com a capacidade de 500kg, que devem ser descarregadas manualmente para dentro do tegão, como representado nas figuras 4 e 5. Portanto os camiões basculantes podem descarregar as uvas muito mais rápido no tegão (uma velocidade de descarga à volta de 2 ton/min) do que os não basculantes (à volta de 500 kg/min).

Figura 4: Camião não basculante - Fonte: [3]

Figura 5: Descarga das uvas de um camião não basculante para o tegão - Fonte: [3]

Os tegões tipicamente têm uma capacidade relativamente maior que os camiões, de à volta de 10 toneladas. No entanto, eles também têm uma velocidade de descarga mais lenta que as prensas. Portanto, um tegão pode geralmente aguentar as cargas de uvas de vários camiões, mas eles são calibrados para processar à volta de 25 toneladas de uvas por hora, ou 416 kg/min. Isto significa que os tegões são o primeiro *bottleneck* deste processo. Não só eles estão em menor número que os camiões, como também o processamento das uvas é mais lento. Isto significa que é frequente que um camião que esteja a descarregar num tegão, tenha que fazer pausas quando este fica cheio, ou que este descarrega as uvas a uma taxa que não é a óptima.

Em relação às prensas, estas poderão ter diferentes capacidades, mas são geralmente muito maiores que os tegões (ex. 25 a 50 toneladas). Cada prensa deve receber, geralmente, uvas de vários tegões. Portanto, ao contrário da transição de camião para tegão, não existem *bottlenecks* enquanto a prensa está a encher. No entanto, assim que a prensa começa o processo de esmagamento, esta pára de receber uvas. Como o ciclo de cada prensa demora 4 horas (independentemente da capacidade da prensa), esta fase é o maior *bottleneck* deste processo. Portanto, as prensas devem preferencialmente estar na sua capacidade máxima quando iniciam o processo de esmagamento, sendo que nem sempre isto é possível.

O principal objetivo de todo o processo é portanto gerir os camiões e prensas de forma a

otimizar o carregamento de cada prensa e minimizar o tempo de espera das uvas, enquanto são priorizadas as uvas mais frágeis. Isto é, que camião vai para que tegão a qualquer momento, para que prensa deve cada tegão enviar as suas uvas, e a que momento a prensa começa o seu ciclo.

A principal restrição é que a adega geralmente lida com diferentes variedades de uvas (ex. Loureiro, Fernão Pires) e muitas delas não podem ser processadas em conjunto. Isto depende de fatores como taxa de álcool da variedade da uva, ou as características desejadas do vinho que será produzido. Portanto, dependendo do número de camiões em espera, os condutores podem ter que esperar de 3 a 4 horas até que seja possível a descarga dos seus camiões. A satisfação dos condutores é outro fator importante no processo, que frequentemente colide com os objetivos da adega. De facto, geralmente uma estratégia *first-in-first-out* (ou seja, o primeiro a entrar é o primeiro a sair) não é a melhor do ponto de vista da produção, e alguns condutores podem ter que esperar muito mais que outros, dependendo da variedade de uvas que têm e o estado da adega na sua chegada. A satisfação dos condutores pode ser medida através de variáveis como tempo médio de espera individual.

Externamente, o problema também tem algumas restrições. Primeiro, as uvas são colhidas num intervalo de tempo relativamente curto, normalmente entre os meses de setembro e outubro. Durante cada dia deste período, é muito difícil prever, e muito mais controlar, quando as uvas irão chegar. A colheita das uvas e a sua chegada depende da interseção de fatores meteorológicos e biológicos, assim como intuição e experiência dos produtores que decidem quando a colheita deve ser feita. Além disso, a chegada das uvas nos dias de colheita não segue uma distribuição uniforme durante o dia, visto que existem períodos em que existe a chegada de um grande número de camiões. Isto deve-se ao facto de que todos os produtores tendem a colher as suas uvas de manhã, portanto os camiões tendem a começar a chegar ao fim da tarde. Isto significa que existem momentos em que a adega está a operar bem abaixo da sua capacidade máxima, enquanto que em outros momentos existem longos períodos de espera. Devido ao facto da adega não ser capaz de controlar a periodicidade de momentos em que as uvas são entregues, o foco deve ser colocado na otimização do processo depois dos camiões terem começado a chegar, de acordo com as características e quantidade das suas uvas e máquinas disponíveis no momento.

De modo a que ao longo e no final deste projeto fosse possível realizar a descrição detalhada do problema, simulações de um cenário real e analisar os resultados das mesmas, foi necessário realizar alguma pesquisa sobre potenciais grandes empresas vinícolas que pudessem fornecer alguns dados sobre as mesmas e sobre as suas colheitas. Várias foram contactadas, sendo que a seleção passou pela Aveleda e Sogrape devido à disponibilidade e rápida resposta.

A Aveleda é uma empresa familiar constituída em 1870, dedicando-se à cultura da vinha e

do vinho. Com sede na Quinta da Aveleda, inserida na Região dos Vinhos Verdes, a propriedade vitivinícola adotou o nome donde provêm as uvas, Aveleda, produzindo igualmente vinhos das regiões da Bairrada e Douro. De destacar ainda o prémio Best of Wine Tourism conquistado em 2011 pela Quinta da Aveleda [12]. Foi realizada uma visita às instalações da Aveleda e os dados fornecidos pela mesma foram usados maioritariamente na descrição do problema apresentado.

A Sograpé é uma empresa com sede em Avintes, Portugal que se dedica a produzir vinhos de elevada qualidade e grandes marcas. Ela distribui os seus vinhos na Europa, na América, em África e na Ásia, fazendo-os chegar a mais de 120 países, estando entre eles marcas internacionalmente relevantes [13]. Foi feito um pedido para saber se seria possível o fornecimento de dados relativos à mesma, sendo que estes foram prontamente enviados, o que contribuiu para que esta simulação fosse realizada com ainda mais rigor e com um cenário real.

A Sograpé, mais especificamente o Centro de Vinificação de Anadia da Sograpé, durante as vindimas deste ano, 2019, registou 1998 camiões a entrarem na adega, durante o período de colheitas (época alta) que durou de 12 de setembro a 7 de outubro, ou seja 17 dias. Em média foram recebidos 117 camiões por dia, cada um com uma carga média de 2,341 toneladas, sendo que durante este período foram recebidas 4678 toneladas de uvas. Esta é uma das maiores adegas da Sograpé, sendo um ótimo exemplo para analisar o comportamento do Algoritmo Genético proposto. Na Secção 4.4 são descritos todos os detalhes acerca da empresa.

1.2 Objetivos

Este projeto tem como objetivo estudar e utilizar algoritmos de otimização para resolver o problema de receção de uvas em adegas durante o período de vindimas como descrito na Secção anterior.

Para isso, foram escolhido utilizar Algoritmos Genéticos (GA), tendo sido desenvolvido um GA capaz de, através de um conjunto de *inputs* iniciais, encontrar a melhor solução para a otimização do processo de receção e processamento das uvas de uma adega.

Para este projeto um conjunto de objetivos foi considerado, sendo estes os seguintes:

- O algoritmo deve ser capaz de realizar a simulação de cenários o mais próximo do tempo real possível, deste modo os utilizadores poderão tomar decisões tão rápido quanto possível;
- Minimizar o tempo de espera das uvas em cada um dos camiões dos fornecedores da adega, devendo ser melhores do que atualmente existentes nas adegas (num cenário real)
- Diminuir do tempo de espera dos produtores;

- Otimizar a gestão dos recursos disponíveis na adega;
- Reduzir do tempo de processamento das uvas, de modo a que haja a garantia de uma melhor qualidade dos vinhos produzidos, visto que é minimizada a deterioração das mesmas ao longo do tempo;
- Conseguir obter dados dados reais de uma empresa e simular um cenário, de modo a comparar resultados e a verificar as verdadeiras vantagens que este algoritmo trará para as empresas.

1.3 Estrutura do documento

De modo a que haja uma explicação clara da estrutura do documento, esta Secção destina-se a esse efeito, sendo este organizado da seguinte forma.

Na Secção 2 é feita uma análise geral sobre os algoritmos inspirados na natureza, sendo introduzido alguns conceitos essenciais dos mesmos. Em 2.1 é feita uma explicação sobre o que são os algoritmos NABI, sendo que após a mesma vários exemplos serão analisados. Para cada um deles é explicada qual a inspiração por trás do algoritmo, como é que ele funciona, concluíndo com a sua explicação com vários exemplos de aplicações reais. Em 2.2 o estado da arte é concluído, sendo feita uma análise geral do que é referido nas secções anteriores e dada a explicação sobre a escolha do GA para a execução deste projeto.

De seguida, na Secção 3 é explicado, tendo em conta uma representação básica do problema, como foi feita a modelação da solução, e como foi implementada cada um dos componentes do GA, nomeadamente a inicialização, função de fitness, reprodução e seleção.

Posteriormente na Secção 4 são descritas as três abordagem implementadas: FIFO, representado o comportamento real/atual das adegas, a aborgagem GA implementada e por fim a simulação e análise do comportamento do GA com dados reais.

O documento termina com a Secção 5 em que é feita a revisão geral o projeto, quais as vantagens do GA, quais os objetivos atingidos através da utilização do mesmo e trabalho futuro.

2 Estado da arte

Algoritmos inspirados na natureza têm sido objeto de muitos estudos de várias áreas cientificas, devido à sua elevada eficiência em resolver problemas da vida real. Algoritmos como: algoritmos genéticos, *differential evolution*, *simulated annealing*, *harmony search*, *particle swarm optimization*, *ant colony optimization*, *firefly algorithm* e *bat algorithm*, serão analisados nesta Secção.

A maioria dos problema de pesquisa e otimização são não lineares e apresentam um elevado número de restrições. Consequentemente encontrar soluções para estes problemas requer algoritmos de otimização eficientes [14]. Ao longo das últimas décadas muitos algoritmos foram estudados e implementados de modo a resolver problemas com um alto nível de complexidade. Dois exemplos desses algoritmos são os heurísticos e meta-heurísticos [15]. Devido à elevada complexidade deste tipo de problemas é difícil testar todas as possibilidades/combinações. Utilizando algoritmos de aproximação é possível encontrar, num espaço de tempo relativamente curto, soluções de qualidade para resolver problemas de otimização complexos. No entanto não existe a garantia de que a solução ótima será atingida. Os dois principais componentes de qualquer algoritmo meta-heurístico são: intensificação e diversificação [16, 17]. Diversificação consiste em procurar por soluções promissoras num maior espaço de pesquisa, que ainda precisa ser refinado. Esta operação é equivalente a diversificar a pesquisa de modo a evitar a ficar restrito a um ótimo local (pesquisa global). Estes termos são geralmente usados em contextos genéricos. Intensificação consiste em procurar por uma solução numa região limitada do espaço de pesquisa de modo a que uma solução seja encontrada. Esta solução é equivalente a melhorar a pesquisa nas proximidades da melhor solução encontrada (local search). Normalmente estes termos são usados em técnicas de otimização baseadas em populações (*population-based optimization techniques*).

Grande parte destas meta-heurísticas derivam do comportamento de sistemas naturais e são conhecidos como algoritmos *nature and bio-inspired* (NABI) que de acordo com [18] podem ser organizados em 5 categorias: *evolutionary algorithms*, *physical algorithms*, *swarm intelligence algorithms*, *sbio-inspired algorithms* e outros algoritmos inspirados na natureza. As meta-herísticas NABI têm sido estudadas por uma grande variedade de áreas cientificas [19]. Nesta Secção é apresentada uma revisão da literatura das principais meta-heurísticas inspiradas na natureza, referenciando algumas aplicações das mesmas no setor agrícola.

A revisão do estado da arte está organizado da seguinte forma: A Secção 2.1 apresenta uma breve revisão de alguns algoritmos NABI encontrados na pesquisa bibliográfica, com referência ao mecanismo que inspirou a criação dos mesmos, sendo descrito também o funcionamento dos mesmos e deixado um web site onde uma implementação em MATLAB pode ser descarregada. A

Secção 2.2 conclui o estado da arte.

2.1 Algoritmos NABI

Atualmente existe um elevado número de algoritmos NABI. Nesta Secção os seguintes algoritmos foram selecionados, nomeadamente: Algoritmos Genéticos (GA) [20], Evolução Diferencial (DE) [21], *simulated annealing* (SA) [22], *harmony search* (HS) [23], *particle swarm optimization* (PSO) [24], *ant colony optimization* (ACO) [25], *firefly algorithm* (FA) [26]e *bat algorithm* (BA) [27]. As razões para a seleção destes algoritmos foram as seguintes: (1) a maioria deles são considerados meta-heurísticas que são utilizados numa ampla gama de aplicações bem sucedidas inclusive na agricultura. Este é o caso dos: GA, DE, SA, PSO e ACO; (2) Meta-heurísticas NABI recentes, estão a ser usados para resolver problemas de pesquisa e otimização na agricultura.

Nesta Secção a seguinte metodologia é usada em cada um dos algoritmos: (1) Uma apresentação do mesmo é feita, incluído os seus princípios inspirados na natureza; (2) Algumas aplicações usadas na agricultura são apresentadas.

2.1.1 Algoritmos Genéticos

Charles Darwin desenvolveu a teoria da evolução natural na origem das espécies. Ao longo de várias gerações, organismos biológicos evoluem com base no princípio da seleção natural "sobrevivência do mais apto" para alcançar certas tarefas notáveis. A caça por alimento eficiente dos albatroz, a semelhança entre tubarões e golfinhos e assim por diante, são os melhores exemplos de conquista da evolução aleatória sobre a inteligência. Assim, como funciona tão bem na natureza, como resultado é interessante simular a evolução natural e desenvolver um método que resolva problemas de pesquisa e otimização concretos.

Na Natureza, indivíduos de uma população competem entre si por recursos, como comida abrigo e assim por diante. Também na mesma espécie, os indivíduos competem para atrair parceiros para a reprodução. Devido a esta seleção, indivíduos com baixo desempenho têm menos chance de sobreviver, e os mais aptos, produzem uma maior quantidade de descendência. Deve também ser destacado que durante a reprodução, a recombinação das boas características de cada ancestral pode produzir descendência "mais apta", comparativamente com a dos pais. Após algumas gerações, as espécies evoluem espontaneamente, tornando-se mais e mais adaptadas ao seu ambiente.

Os GA tornaram-se populares através do trabalho de John Holland [20] durante a década de 1980 e através do seu livro *Adaptation in Natural and Artificial Systems* de 1975. O seu trabalho

teve origem através dos seus estudos com autómatos celulares, realizados com a colaboração dos seus estudantes da universidade de Michigan. Holland introduziu uma estrutura formalizada para prever a qualidade da próxima geração, conhecida como *Holland's Schema Theorem*. Pesquisas com GA mantiveram-se apenas teóricas na sua maioria até a meio da década de 1980, quando a primeira conferência internacional em GA foi realizada em Pittsburgh na Pensilvânia. Através do seu livro Holland descreveu como seria possível aplicar os princípios da evolução natural, segundo Charles Darwin, em problemas de otimização e assim construiu o primeiro GA. A teoria de Holland foi desenvolvida ao longo do tempo e atualmente os GA são poderosas ferramentas utilizadas na resolução de problemas de otimização. Os GA são baseados nos princípios da genética e evolução.

Os GA são meta-heurísticas inspiradas pelo processo e seleção natural que pertencem à classe dos algoritmos evolucionários (EA) e são frequentemente utilizados para gerar soluções de qualidade para problemas de pesquisa e otimização. Os principais operadores dos GA são: seleção, crossover e mutação [28]. O processo evolucionário começa com uma coleção, chamada de população inicial. Cada cromossoma representa uma solução para o problema a ser resolvido e é caracterizado por uma coleção de parâmetros (variáveis) conhecidos como genes. No GA original, a coleção de genes é representada por uma string binária. A função de fitness determina o quão apto é um cromossoma (a habilidade de competir com outros cromossomas) e atribui uma pontuação de aptidão que representa a probabilidade de ser selecionado para reprodução. Durante a seleção, o cromossoma mais apto poderá ter a possibilidade de passar os seus genes para a próxima geração, visto que estes têm mais probabilidade de serem selecionados para a próxima geração. O crossover é o operador mais importante dos GA, quando o ponto de interseção entre genes é escolhido aleatoriamente por cada par de pais a juntar. Em certas descendências, alguns dos seus genes podem ser mutados, o que implica que algumas das suas características tenham que mudar. A mutação ocorre para manter a diversidade da população e evitar uma conversão prematura. A execução do GA pode acabar quando convergir, ou seja, quando não produz gerações em que são significativamente diferentes das anteriores, fornecendo assim um conjunto de soluções para um dado problema [29]. Para além da reprodução/crossover e mutação, é possível usar outros operadores, como reagrupamento, *colonization-extinction* ou migração [30]. O diagrama de uma possível implementação de um GA pode ser visualizado na imagem 6.

Uma implementação no MATLAB pode ser descarregada em [31].

Em [32] é descrito um método para a criação de um caminho sub-ótimo de um robô ligado à agricultura usando uma técnica de controlo combinando um GA e uma rede neural (NN). Em [33] apresentada uma metodologia de otimização multi-objetiva para a o criação de uma rede de sensores wireless e gestão energética. Uma aplicação de sensores de precisão ligados à agricultura

foi utilizado como exemplo. Em [34], um GA e uma *Support Vector Machine* (SVM) são propostos para prever o consumo de água na agricultura. A previsão do consumo de água é muito importante para definir o planeamento da otimização da configuração dos recursos hídricos. [35] propõe uma abordagem agrícola inteligente para efetuar a previsão do tempo usando um GA e *Fast Fourier Transform* (FFT). O modelo proposto ajuda os agricultores a planear as suas atividades.

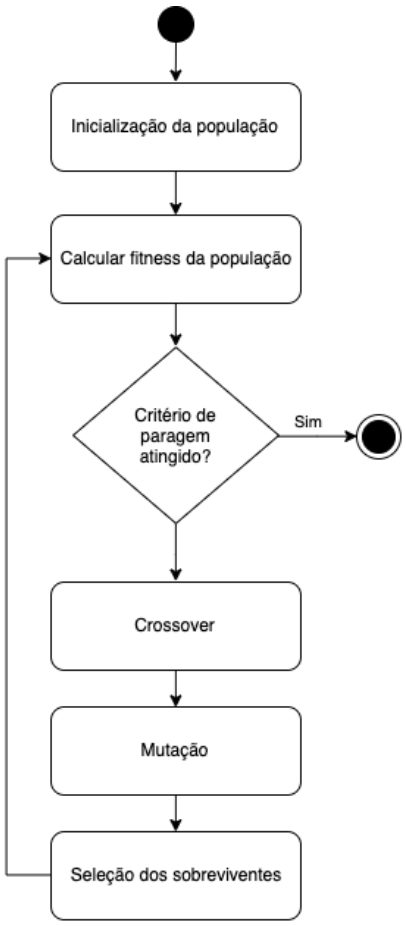


Figura 6: Diagrama genérico de um possível fluxo de um GA

Em [36], a atividade das bombas de água para os sistemas de irrigação, limitada pela quantidade de energia disponível, foi simulada e otimizada de forma a minimizar o consumo de energia. [37] propõe um particionamento automatizado da terra usando um GA, mostrando que é possível acelerar o processo de particionamento. A terra particionada é um dos maiores problemas no que diz respeito ao processo de consolidação da terra sendo o processo mais importante para prevenir a fragmentação da terra e garantir a produtividade agrícola.

2.1.2 Harmony Search

Harmony Search (HS), apresentado por [23] em 2001, é baseado em conceitos musicais e pode ser inserido na classe de meta-heurísticas inspiradas na física. HS é uma meta-heurística que imita fenómenos e é inspirada no processo de improvisação de músicos de jazz [23]. É feita uma analogia entre o processo de improvisação de uma banda de jazz e um processo de otimização. Músicos tentam encontrar uma harmonia que é determinada por um padrão estético, da mesma forma que um processo de otimização tenta procurar por uma solução ótima, que é determinada por uma função objetivo [38]. De uma forma simples, cada músico reproduz uma nota (um valor) para encontrar a melhor harmonia (ótimo global). Na natureza, a harmonia é definida por uma relação especial entre várias ondas sonoras que possuem diferentes frequências. A qualidade da harmonia improvisada é determinada por uma estimativa estética. De modo a melhorar a estimativa estética, os músicos praticam ao longo do tempo.

Existem semelhanças entre a improvisação musical dos músicos e processos de otimização. Num problema de otimização, o maior objetivo é encontrar o ótimo global da função objetivo em consideração, ajustando um número predefinido de variáveis de decisão. De fato, num problema de otimização, as variáveis de decisão originam um vetor de soluções. Portanto, os valores das variáveis de decisão são usados na função objetivo e a qualidade do vetor de soluções é calculada. O vetor de soluções é atualizado ao longo de cada iteração até que o ótimo global seja obtido.

A comparação entre a música e os processos de otimização revelam as seguintes semelhanças [39]:

- Num processo musical, a qualidade da harmonia é determinada pela sua estimativa estética. No processo de otimização a qualidade do vetor de soluções é determinada pelo valor da função objetivo
- No processo musical, o objetivo final é obter a melhor harmonia, enquanto que no processo de otimização, o objetivo final é obter o ótimo global.
- No processo musical, os músicos mudam o tom dos seus instrumentos. Um algoritmo de otimização muda os valores das variáveis de decisão.
- No processo musical, qualquer tentativa de tocar uma harmonia é chamada de prática. Numa otimização, cada tentativa de atualizar o vetor de soluções é chamada de iteração.

Em geral, quando um músico deseja afinar seu instrumento (por exemplo, violino, saxofone, etc) e tocar uma nota, ele/ela utiliza uma das três maneiras possíveis. Estas regras são o corpo principal do algoritmo HS.

1. Ele/ela pode tocar uma nota aleatoriamente, dentro do intervalo possível.
2. Ele/ela pode tocar uma nota da sua própria memória.
3. Ele/ela pode tocar um nota próxima da nota escolhida pela sua memória.

No processo de música, a qualidade da harmonia é determinada pelo tom de cada instrumento, que é o valor da função objetivo é determinado por um conjunto de valores atribuídos a cada variável. O músico faz várias tentativas para encontrar a harmonia, assim como o algoritmo HS executa várias iterações até encontrar a melhor solução. Em suma, verificou-se que ambos os processos deveriam ser ideais em suas áreas [40]. O HS é simples em conceito, com poucos parâmetros, e de fácil implementação [41], mas muito eficiente. Como descrito em [23], um conjunto de soluções (memória da harmonia) é gerada aleatoriamente. Dado um conjunto de soluções uma nova solução é gerada e se esta for melhor que a pior solução presente na memória da harmonia, a pior solução é substituída pela nova. O processo é repetido até que o critério de paragem seja satisfeito. Este fluxo pode ser visualizado na Figura 7.

Uma implementação em MATLAB pode ser descarregada em [42].

Em [43], uma abordagem HS é proposta para otimizar o problema de planeamento do caminho de cobertura (CPP) que é aplicado a veículos aéreos, com intenção de minimizar o tempo dos turnos, para garantir que o tempo da missão seja igualmente minimizado. Em [44], um sistema baseado num HS foi desenvolvido permitindo prever o comprimento da parte aérea de uma planta de mostarda. De acordo com este estudo, medir o crescimento da planta em relação ao comprimento da parte aérea é o método mais conveniente para estimar a produção da planta.

Sendo um poderoso algoritmo de otimização, o HS tem atraído muita atenção para resolver vários tipos de problemas de otimização durante os últimos anos. As vantagens do algoritmo HS são a sua fácil implementação, conceito simples e reduzido número de parâmetros necessários para a sua configuração. Devido a estas vantagens, o HS tem sido usado para resolver vários problemas em várias áreas como sistemas de energia, comunicação, software e reconhecimentos de padrões [39]. O estudo da literatura [39] indica que os HS podem resolver eficientemente vários tipos de problema de otimização. Várias variantes do algoritmo HS podem ser encontradas com o objetivo de melhorar o processo de improvisação e configuração de parâmetros. Existem outras variantes que tentam melhorar a performance dos HS, retirando algumas ideias de outra metaheurísticas como GA, PSO e DE. Tendo em conta os resultados das aplicações do HS referenciadas anteriormente e as vantagens do mesmo, é possível dizer que este é um bom candidato para a resolução de problemas de otimização complexos.

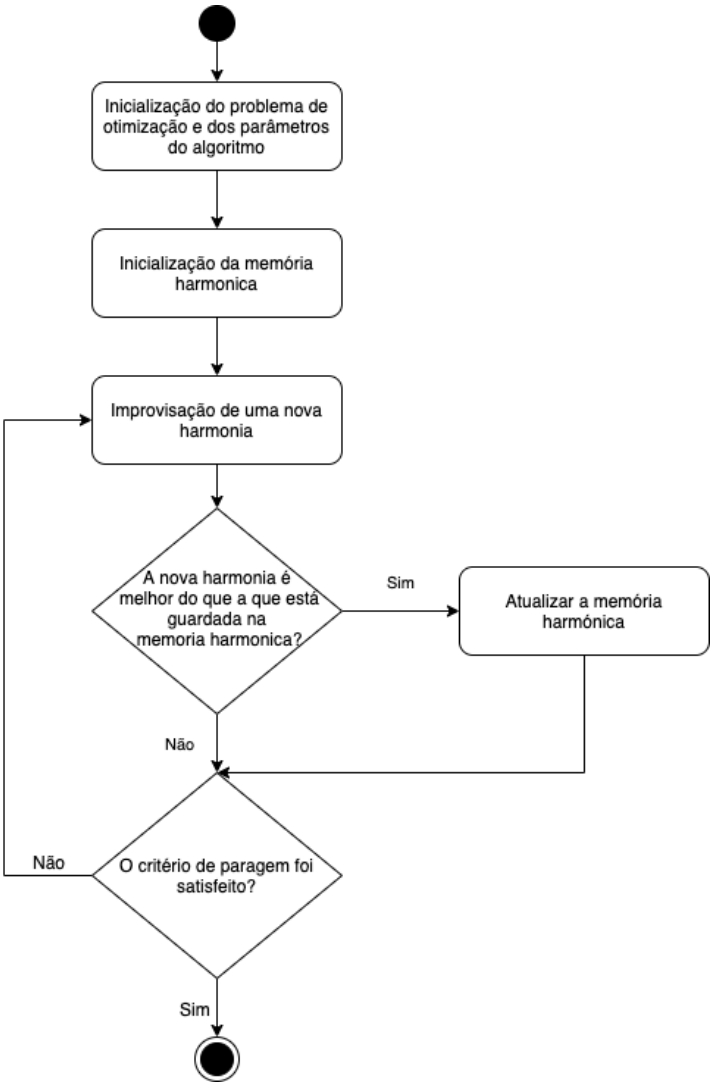


Figura 7: Diagrama genérico de um possível fluxo da *Harmony Search*

2.1.3 Simulated Annealing

O algoritmo *simulated annealing* (SA) foi proposto por [22] em 1983. É uma técnica probabilística inspirada por processos de tratamento térmico na indústria metalúrgica [45]. O processo metalúrgico consiste em aquecimento e arrefecimento controlado do material, alterando, assim, certas propriedades físicas, permitindo por exemplo eliminar defeitos no material [46]. O aquecimento é feito a uma temperatura acima da temperatura de recristalização do metal, seguido por um lento e gradual arrefecimento. O SA aceita pesquisas para soluções piores (menor qualidade) que permitem algumas subidas que permitem também escapar dos mínimos locais [47]. A aprovação de soluções piores é uma propriedade fundamental das meta-heurísticas, porque permitem uma

pesquisa mais ampla pela solução ideal. Nos SA, o arrefecimento lento é interpretado como uma diminuição lenta na probabilidade de aceitação de soluções piores enquanto é feita a pesquisa por soluções. O algoritmo SA começa com um única população inicial. Depois uma solução vizinha é selecionada aleatoriamente e o custo de ambas as soluções é calculado. Se a nova solução tiver um custo menor então o algoritmo aceita-a, caso contrário ela aceita-a com a probabilidade $e^{-\Delta/T}$ onde Δ representa a diferença entre o custo da solução e T é a temperatura. Inicialmente T assume um valor alto predefinido e é reduzido progressivamente a cada iteração. O critério de paragem pode ser a temperatura chegar a zero [48]. A representação genérica do SA pode ser visualizada na Figura 8.

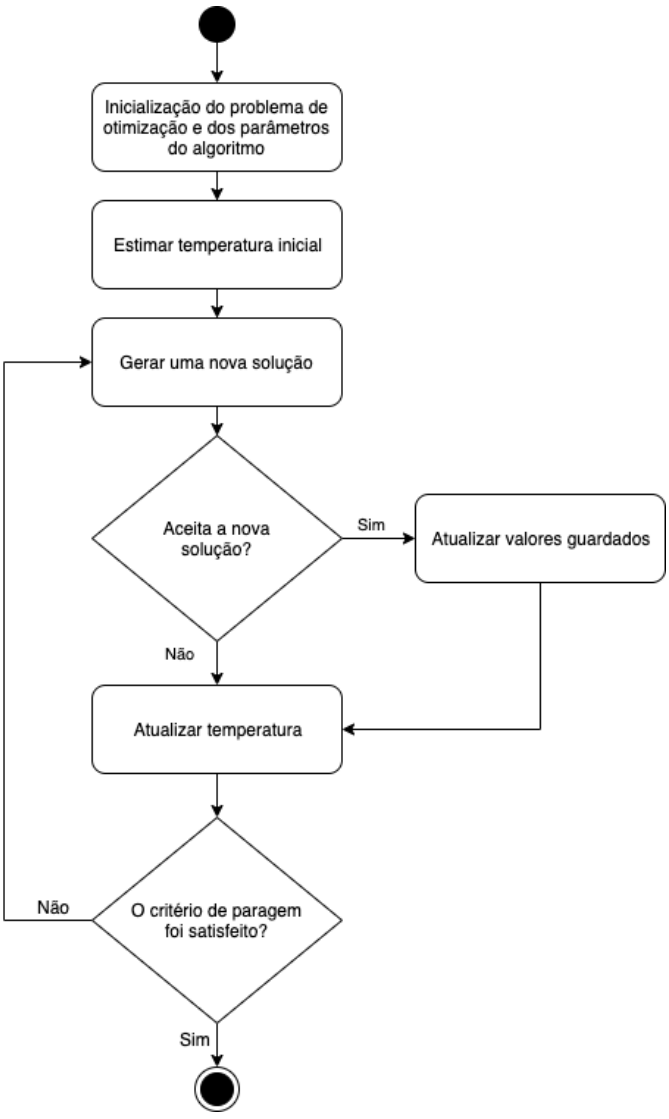


Figura 8: Diagrama genérico de um possível fluxo de um algoritmo SA

Uma implementação em MATLAB pode ser descarregada em [49].

[50] propõe uma nova metodologia de maximização usando técnicas SA para recuperar energia, considerando a viabilidade da bomba funcionar como turbina em redes de irrigação. As técnicas de SA foram usadas com diferentes funções objetivo, assim como diferentes números de máquinas. O estudo apresentado em [51] tenta caracterizar os padrões espaciais do chumbo (Pb) no solo de campos de arroz, comparando a simulação sequencial de Gauss, técnicas SA e métodos *Kriging*. [52] aborda o potencial do uso de binóculos baseados em visão estéreo para a análise tridimensional de plantas únicas e estimativas de atributos geométricos como altura e total da área das folhas. Para executar a correspondência estéreo, um método SA foi usado para encontrar os melhores candidatos para a correspondência, levando em consideração os *pixels* vizinhos. [53] apresenta um método inovador de apoio à decisão do agendamento da irrigação de uma fazenda, chamado *Canterbury Irrigation Scheduler* (CIS) que é adequado quando a disponibilidade sazonal de água é limitada. As estratégias de irrigação são definidas por um conjunto de variáveis de decisão e otimizadas usando um SA.

2.1.4 Evolução Diferencial

A primeira publicação da evolução diferencial (DE) foi relatada por [21] em 1995. O DE é similar ao GA, visto que ambos são baseados nos princípios evolucionários da biologia, como mutação, crossover e seleção. O DE é um algoritmo baseado na evolução que nativamente suporta representações de soluções baseadas em *floanting-point* [54]. De acordo com [55], o DE tem uma estrutura simples, fácil de usar e com uma velocidade e robustez considerável, tornando-os um dos melhores algoritmos baseados na evolução para resolver problemas do mundo real. O DE é uma técnica de pesquisa baseada num vetor populacional (*target vector* - contém a solução atual; *mutant vector* - corresponde à mutação do do vetor alvo; *trial vetor* - vetor resultante depois da troca de operações entre os vetores *target* e o *mutant*). A ideia base por trás do DE é uma nova estrutura para gerar vetores de parâmetros de teste. O DE gera um novo vetor de parâmetros adicionando o vetor de diferença ponderada entre dois membros da população a um terceiro membro. Se o vetor resultante produzir um valor mais baixo na função objetivo do que outro membro da população, o vetor que acabou de ser gerado é substituído pelo que foi comparado com ele. Em adição, o melhor vetor de parâmetros é avaliado em cada geração de modo a verificar o progresso durante o processo de otimização. Extraindo a informação sobre a distância e direção da população para gerar resultados aleatórios numa estrutura adaptativa com excelentes propriedades de convergência [56]. Uma implementação em MATLAB do DE pode ser descarregada em [57].

Os modelos agrícolas podem frequentemente ser associados como problemas de otimização

complicados de resolver. Em [58] um algoritmo DE multi-objetivo é apresentado através de quatro estratégias para resolver um modelo de planeamento de colheitas com múltiplas restrições. Os objetivos do modelo são minimizar o consumo de água para a irrigação e maximizar o total de colheitas colhidas, assim como o total da produção agrícola. Em [59] um algoritmo DE é usado para resolver o problema do planeamento de cultivo. Os três objetivos propostos para a resolução do problema de planeamento do cultivo são a maximização dos lucros totais líquidos e o *output* total da colheita, enquanto é minimizado o total de água usada para a irrigação.

2.1.5 *Particle Swarm Optimization*

As ideias iniciais de Kennedy (um psicólogo) e Eberhart (um engenheiro electrotécnico) eram essencialmente destinadas a produzir inteligência computacional explorando simples analogias com a interação social, em das habilidades cognitivas de cada indivíduo. As primeiras simulações (Kennedy e Eberhart em 1995) [24] foram influenciados pelo trabalho de Heppner e Grenander em 1990 e utilizaram analogias com o comportamento de bandos de pássaros à procura de milho. Pouco tempo depois eles desenvolveram um método de otimização apelidado de *Particle Swarm Optimization* (PSO) [60].

O PSO tem algumas capacidades de pesquisa, no entanto, as partículas lidam com um problema: explorar espaços existentes já ocupados, ou explorar novos e desconhecidos espaços de pesquisa. A exploração global pura reduz a precisão do cálculo, mas é propícia a encontrar possíveis novas soluções, por outro lado a exploração local pode refinar soluções existentes, mas frequentemente cai na convergência de soluções locais.

Encontrar o ótimo global em muitas funções complexas acarreta vários desafios e até mesmo situações impossíveis [61]. No entanto, ao longo das últimas 2 décadas, existem muitos relatos na literatura que descrevem melhoramentos no desempenho dos PSO, tais como diferentes configurações dos parâmetros de controlo [62], incorporação de uma estratégia híbrida [63, 64], estratégias cooperativas multi enxame [65], diferentes redes topológicas sociais [66], diferentes estratégias de aprendizagem social [67], entre outros.

Um equilíbrio entre a exploração local e global é crucial para melhorar a eficiência de um PSO [68]. Apesar de muitas estratégias em diferentes algoritmos de otimização lidarem com este dilema com alguma extensão, espera-se que novos estudos tenham um melhor desempenho [63]. A exploração da natureza intrínseca do algoritmo pode ser a resposta para isso. Transformar a natureza do PSO ou das suas variantes, traria ainda mais vantagens para a resolução de uma grande variedade de problemas de otimização.

A PSO tem atraído a curiosidade de vários investigadores por todo mundo. É uma técnica

de otimização estocástica [69], inspirada pelo comportamento social de conjuntos de animais (ex. bandos de pássaros e cardumes de peixes). De uma forma simples, o PSO otimiza o problema tentando melhorar iterativamente soluções candidatas contra um dada medida de qualidade. Potenciais soluções, chamadas de partículas, deslocam-se através do espaço de pesquisa de acordo com algumas fórmulas simples [70], onde o movimento de cada partícula individual é influenciado pela sua posição mais conhecida, mas também pelas melhores posições conhecidas de outras partículas. Isto faz com que o enxame se mova iterativamente para as melhores soluções, como representado na Figura 9.

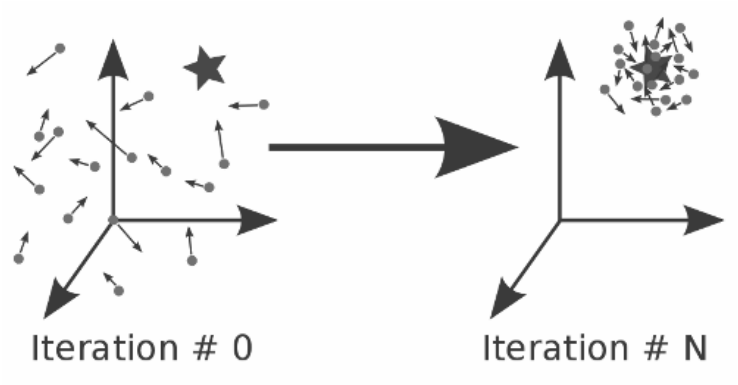


Figura 9: Representação de um resultado de um algoritmo PSO [4]

Uma implementação em MATLAB pode ser descarregada em [71].

Em [72], uma rede neural RBF baseada em PSO é proposta para fundir informação dada por múltiplos sensores de posição para obter uma informação mais precisa sobre a navegação de veículos agrícolas. Em [73], o PSO e o SVM são combinados para a previsão de consumo de água na agricultura, que é importante para otimizar a configuração dos recursos hídricos. [74] apresenta um sistema baseado no PSO para resolver o planeamento de rotas dos veículos de colheita de cana-de-açúcar. Uma nova estrutura de *encoding/decoding* da nova partícula foi desenvolvida para combinar o planeamento do caminho com a acessibilidade e as restrições de colheita divididas. Em [75], um PSO é proposto como um novo método para desenvolver um controlador de temperatura do ar sujeito a restrições. As saídas do controlador são compostas para otimizar o comportamento futuro das variáveis do ambiente da estufa.

O real potencial da PSO deriva das interações entre a colaboração cooperativa entre partículas no espaço de pesquisa. Se a colaboração entre partículas for removida do algoritmo, a performance do mesmo é abismal, como referido em [69], que mostra que ao longo da pesquisa do PSO, os indivíduos são levados em direção ao sucesso uns dos outros, tendo como resultado frequente a

aglomeração de indivíduos em regiões ótimas do espaço de pesquisa.

2.1.6 Otimização da colónia de formigas

A inteligência de enxames tem sido uma abordagem usada para resolver problemas e que se inspira em comportamentos sociais de insetos e de outros animais. Em particular, as formigas têm inspirado vários métodos e técnicas e dentro destes, um dos mais bem sucedidos é a otmização da colónia de formigas ou *ant colony optimization* (ACO).

A ACO é uma abordagem meta-heurística baseada em populações. Foi proposta por Marco Dorigo para resolver vários problemas de otimização discretos. O algoritmo ACO genérico imita a forma como formigas reais encontram o caminho mais curto entre a comida e o seu ninho. As formigas comunicam entre si através de caminhos que são preenchidos com feromonas e através destes trocam informação indiretamente qual caminho deve ser seguido. Caminhos com maior concentração de feromonas terão uma maior chance de serem seguidos e assim reforçados posteriormente, enquanto que a concentração de feromonas dos caminhos que não são escolhidos diminui ao longo do tempo através da evaporação das mesmas, o que reduz a atratividade dos mesmo para com outras formigas [76], como pode ser verificado na Figura 10.

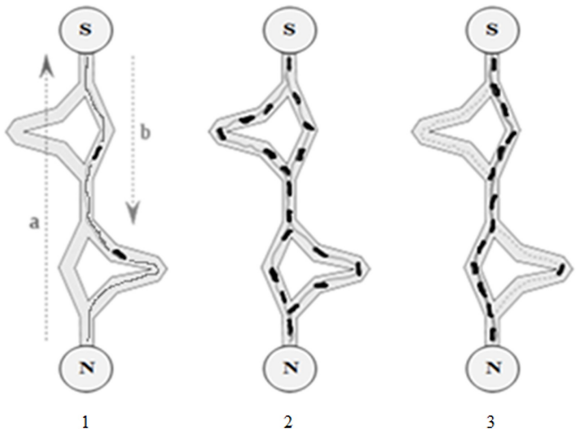


Figura 10: Evolução da escolha do caminho mais curto pela colónia de formigas [5]

Esta evaporação tem a vantagem de evitar a convergência para uma solução óptima local, porque se não existisse evaporação, os caminhos escolhidos pelas primeiras formigas tornar-se-ão cada vez mais atrativos à medidas que as seguintes formigas passassem no mesmo e nesse caso a exploração do espaço de pesquisa seria limitada [77]. Esta forma de comunicação indireta é conhecida como estigmergia e permite à colónia de formigas a possibilidade de encontrar o caminho mais curto. O primeiro algoritmo que seguiu os principios da meta-heurística ACO é o *Ant System*

(AS), proposto por Marco Dorigo em 1996, onde as formigas constroem soluções iterativamente e adicionam feromonas aos caminhos correspondentes a estas soluções. A seleção do caminho é um procedimento baseado em dois parâmetros, a feromona e os valores das heurísticas. O valor das feromonas indica o número de formigas que escolheram aquele caminho recentemente, enquanto que o valor das heurísticas depende do problema em si e é diferente para diferentes casos. Devido ao facto do ACO genérico ser facilmente estendido com outros problemas de otimização, várias variantes do mesmo foram propostas, tais como *Ant Colony System* [25], *rank-based Ant System* [78], e *Elitist Ant System* [79]. E as variantes referidas anteriormente foram aplicados em diferentes problemas, como encaminhamento de veiculos [80], agendamento [81] e o problema do caixeiro-viajante [82]. Mais recentemente as formigas começaram a ser usadas para data mining para agrupamento [83] e tarefas de classificação [84]. Uma implementação em MATLAB do ACO pode ser descarregada em [85].

Em [86] uma *framework* genérica de simulação e otimização para o planeamento da irrigação e fertilização foi proposto. O problema é representado sob a forma de um conjunto de grafos de árvores de decisão e a ACO é usada como mecanismo de otimização. [87] propõe um algoritmo de planeamento de tarefas e um modelo para melhorar o processo de contratação de agricultores na agricultura de pequena e média escala. Na agricultura a esta escala, geralmente não é viável que os agricultores comprem equipamentos caros para realizar as tarefas agrícolas, e, portanto, é comum que os agricultores que possuem esse equipamento trabalhem em fazendas vizinhas.

A inteligência de colónias tem sido aplicada com sucesso em vários domínios. As formigas da colónia artificial são capazes de gerar caminhos progressivamente mais curtos, usando informação acumulada sob a forma de um caminho de feromonas. A otimização da colónia de formigas já foi adaptada para ser usada em muitas áreas, desde problemas relacionados a grafos até problemas relacionados à medicina.

2.1.7 Algoritmo do Morcego

O Algoritmo do Morcego, ou *Bat Algorithm* (BA) é uma meta-heurística inspirada na natureza desenvolvida por Yang em 2010 [27] e grandes melhoria têm sido realizados desde então.

Os morcegos são animais interessantes e a sua maior característica é a eco-localização. Característica esta que atraiu o interesse de estudiosos de várias áreas. O mecanismo de eco-localização funciona como uma espécie de sonar: os morcegos, principalmente os micro-morcegos, criam um pulso de som alto e curto e conseguem saber a distância a que estão de um objeto usando as repetições de eco que retornam aos seus ouvidos [88]. Este método de posicionamento notável faz com que os morcegos sejam capazes de distinguir entre um obstáculo e uma presa, permitindo aos

mesmos caçar na escuridão absoluta [89], como representado na Figura 11.

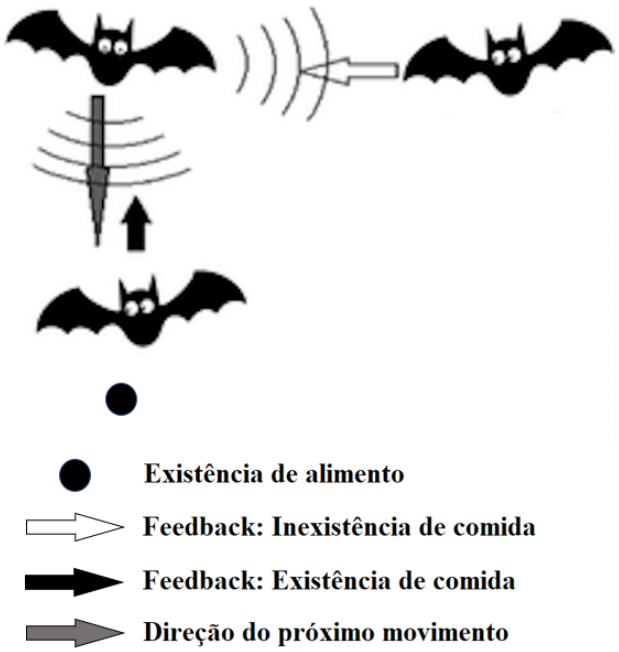


Figura 11: Figura representativa do comportamentos dos morcegos [6]

Motivado pelo comportamento dos morcegos, Yang [90] sugeriu uma nova meta-heurística, o BA. Este foi desenvolvido de modo a comportar-se como um bando de morcegos em busca de presas/comida usando a usa habilidade de eco-localização e tem mostrado uma eficiência promissora na resolução de problemas de otimização globais [15]. Este processo de eco-localização pode ser resumido pelas três regras seguintes [91, 92]: (1): Todos os morcegos usam a eco-localização para determinar a distancia a que estão dos obstáculos no ambiente envolvente, e eles têm também a habilidade de distinguir entre comida/presas e obstáculos; (2): Para procurar por presas, os morcegos $(i = 1, 2, ..., n)$ voam aleatoriamente para uma posição (solução) x_i a uma velocidade v_i com uma frequência fixa de f_{min} e uma sonoridade A_0. Ao procurarem suas presas, os morcegos ajustam automaticamente o comprimento de onda (frequência) e a taxa de emissão de pulsos $r \in [0, 1]$, dependendo da proximidade do alvo; (3): Embora a sonoridade possa variar de várias maneiras, supõe-se que a sonoridade varie de um grande (positivo) A_0 para um valor constante mínimo de A_{min}.

O BA usa essencialmente uma técnica de ajuste de frequência para controlar o comportamento dinâmico de um conjunto de morcegos e balanceia automaticamente a *exploration* (saltos de longo alcance em todo o espaço de pesquisa global para evitar ficar preso no máximo local) com *exploitation* (procure mais detalhes sobre boas soluções para encontrar máximos locais), controlando o

volume e as taxas de emissão de pulsos de morcegos simulados no espaço de pesquisa multidimensional [27]. Uma implementação em MATLAB do BA pode ser descarregada em [93].

De modo a melhorar a ideia original do BA, muitas estratégias foram usadas desde a sua criação, sendo criadas várias variantes do mesmo. Em [94], os autores propõem uma abordagem de agrupamento baseada em BA para resolver problemas de classificação de tipos de culturas usando imagens de satélite multi-espectrais. De acordo com os autores, toda a informação pode ser usada na melhoria geral do rendimento agrícola. Já em [95] Zhang e Wang propuseram um novo BA para processamento de imagens. Eles fizeram duas modificações ao BA original. Primeiramente, usaram uma frequência e sonoridade fixas, de seguida adicionaram um operador de mutação de modo a aumentar a diversidade da população. Eles testaram esta abordagem no processamento de uma imagem e descobriram que o algoritmo que tinham desenvolvido produzia melhores resultados que o BA original. O BA híbrido padrão [96] com técnicas de evolução diferencial. Esta hibridação aprimorou a capacidade de pesquisa local do BA original. Um último exemplo seria o de Xie et al. citado em [97], em que a baixa precisão e velocidade de convergência do BA foram melhoradas. Eles introduziram a trajectória de voos o que aumentou a diversidade de população, por isso o algoritmo efetivamente saiu dos mínimos locais. Eles também usaram o operador diferencial para acelerar a velocidade de convergência. O algoritmo proposto foi testado com várias funções objetivo e concluíram que a abordagem tinha melhores capacidades de aproximação em espaços dimensionais de larga escala.

2.1.8 Algoritmo do Pirilampo

O Algoritmo do Pirilampo, ou Firefly Algorithm (FA) foi desenvolvido por Xin-She Yang em 2008 [26] na Universidade de Cambridge. É um algoritmo de otimização baseado no comportamento dos pirilampos. Este algoritmo é inspirado pelos padrões de luz e comportamento dos pirilampos, conhecidos pela sua bioluminescência. Ele é baseado, portanto, na habilidade dos pirilampos de variarem a intensidade da emissão de luciferino que faz com que eles brilhem a diferentes intensidades. Atualmente é sabido que este mecanismo está associado a três fatores principais: atração de presas, defesa contra predadores e acasalamento. Pirilampos de diferentes espécies têm diferentes tons e padrões de luz. Por meio dessas características, é possível que um pirilampo reconheça outros pirilampos da mesma espécie e escolhe o melhor parceiro, dependendo da intensidade da luz emitida por ele. A luz também pode ser usada para atrair presas, isto é, se um pirilampo estiver com fome, a luz irá brilhar mais intensivamente de modo a tornar a atração de insetos mais eficientes [98].

O FA começa com uma população inicial de pirilampos. A qualidade da solução atual é ava-

liada, a população de pirilampos é classificada de acordo com os seus valores físicos e o melhor indivíduo da população é selecionado. No fim, todos os pirilampos são movidos em direção ao indivíduo mais atrativo, como demonstrado na Figura 12. O processo é repetido até que o critério de paragem seja satisfeito [99]. Em [100] foi introduzida uma nova versão do FA para aplicações de otimização multi-modal. Para simplificar a descrição do algoritmo as três regras seguintes foram usadas: todos os pirilampos são bissexuais e, neste sentido, um pirilampo pode ser atraído por qualquer outro pirilampo, independentemente do género; A atratividade é proporcional à intensidade de luz emitida pelos pirilampos, portanto será sempre o menos brilhante a mover-se em direção ao mais brilhante. Se não existir um pirilampo mais brilhante que o outro, eles movimentar-se-ão aleatoriamente. No entanto, a intensidade (brilho aparente) diminui quando a distância mútua aumenta; O brilho deve ser medido com a função objetivo e, para o problema de otimização, ele pode ser simplesmente proporcional ao valor dessa função. Uma implementação em MATLAB do FA pode ser descarregada em [101].

Figura 12: Figura representativa do comportamentos dos pirilampos [7]

O FA tem duas grandes características: divisão automática e a habilidade de lidar com múltiplas funções objetivo [102]. Isto significa que toda a população de pirilampos pode a qualquer momento se dividir em enxames mais pequenos e cada grupo pode se aglomerar em torno de ótimos locais e, consequentemente a solução global pode ser encontrada entre estes. Para além disso, se o número de agentes de pesquisa é maior que o número ideal, então a propriedade de divisão facilita os agentes de pesquisa a encontrar todos os ótimos ao mesmo tempo.

Os FA mostraram resultados promissores quando aplicados a problemas de otimização contínuos [103]. Yang formulou um FA usando a estratégia de movimentação de voos de Levy e seu desempenho é superior ao GA e PSO [104]. Em [105] distribuição de Gauss é incorporada no FA e permitiu que todos os pirilampos se movessem em direção ao melhor global em cada iteração, o

que aumentou a convergência velocidade do FA. Além disso, em [106] mapas caóticos são usados em combinação com um FA para melhorar o desempenho geral do FA clássico. Adicionalmente, um FA foi desenvolvido para problemas de otimização sem restrições e sua validação foi realizada em várias funções padrão de referência como referido em [107]. No mesmo sentido, Gandomi et al. introduziu o *chaos* com um FA, que é usado para aumentar sua mobilidade de pesquisam, o que permitiu uma otimização global robusta [108]. De modo a melhorar a qualidade das suas soluções, uma nova versão do FA, híbrida, foi desenvolvida com o algoritmo *social spider* [109] e o algoritmo *flower pollination* [110], que resolveram o problema da baixa convergência e aprisionamento a ótimos locais. Um algoritmo FA mutado foi introduzido com base na análise dos movimentos dos pirilampos, usando diferentes probabilidades para cada pirilampo, sendo a mutação realizada em cada um de acordo com sua probabilidade [111]. Em [112], um novo método que usa um FA, com o objetivo de otimizar a operação de um reservatório de abastecimento de água agrícola, é proposto. A função objetivo utilizada foi definida de forma a minimizar a soma das diferenças quadráticas entre as necessidades e as despesas do reservatório divididas pela necessidade máxima durante a operação. O desempenho do modelo usado foi comparado com o desempenho obtido com o GA e o PSO. Em [113], os autores apresentam uma aplicação do FA que estima a utilização de recursos hídricos na cidade de Nanchang na China. Neste caso, o uso destes recursos é dividido em 3 setores: indústria, consumo dos habitantes e agricultura.

2.1.9 Outros algoritmos NABI

Existem outros algoritmos NABI, tais como: *Bacterial Foraging Optimization* [114], *Dendritic Cell Algorithm* [115], *Gravitational Search Algorithm* [116], *Bees Algorithm* [117], *Intelligent Water Drops Algorithm* [118], *Spiral Optimization Algorithm* [119], *Cuckoo Search Algorithm* [120], *Tabu Search Algorithm* [121], *River Formation Dynamics* [122], *Flower Pollina- tion Algorithm* [123] e *Cuttlefish Optimization Algorithm* [124]. Muitos outros podem ser encontrados em [18, 125–127] e em [128] é apresentada uma revisão de vários algoritmos evolucionários e inspirados na Natureza no contexto do controle ambiental de estufas, para um ou vários objetivos, bem como as tendências atuais.

2.2 Análise crítica do estado da arte

Uma introdução às meta-heurísticas inspiradas na natureza e na biologia foi apresentada, seguida da revisão do uso destes maioritariamente em algumas aplicações ligadas ao setor agrícola, mas também em outras áreas. Este conjunto inclui não apenas as mais bem estabelecidas meta-

heurísticas NABI, mas também alguns introduzidos mais recentemente, com aplicações relatadas na agricultura. O número de aplicações encontradas no levantamento bibliográfico sobre os algoritmos GA, DE, SA, HS, PSO, ACO, FA e BA, motivaram a sua revisão mais detalhada. Das aplicações analisadas é possível concluir que os algoritmos GA, DE, SA, PSO e ACO estão presentes em muitas aplicações ligadas à agricultura como referido nos exemplos apresentados na descrição de cada uma deles. Devido ao facto destes serem algoritmos mais antigos e bem estabelecidos era esperado que existissem várias aplicações ligadas à agricultura. No entanto, novos algoritmos como HS, FA e BA também têm algumas aplicações neste contexto e é esperado que o número de aplicações comece a aumentar num futuro próximo devido à crescente adoção de técnicas de precisão na agricultura.

Tendo em conta todos os algoritmos acima descritos, para este projeto os GA foram escolhidos devido à sua fácil implementação, ao facto de não existir a necessidade de realizar muitas configurações para a sua utilização, mas também pelo facto de não existirem documentados, projetos relacionados com este problema utilizando este algoritmo. O facto de não ser necessário existir muitas configurações, é uma vantagem aquando sua utilização por parte do utilizador final, este não irá necessitar de perder muito tempo para o configurar entre cada simulação.

3 Implementação

Numa adega, por vezes é essencial tomar decisões o mais rápido possível de modo que, como descrito num dos objetivos deste projeto, haja a minimização do tempo de espera das uvas. Para isso outro objetivo foi criado, o de desenvolver uma ferramenta capaz de fazer uma simulação de cenários o mais próxima do tempo real possível, visto que como descrito anteriormente, cada segundo conta, e se o utilizador souber quais a melhores opções a tomar, mais rápido todo o processo poderá ser iniciado.

O problema do melhoramento do processo de receção das uvas, como conceptualizado neste trabalho, pode ser visto como um de otimização. De facto, o objetivo é escolher as melhores decisões a qualquer momento havendo uma preocupação com a gestão das máquinas, de acordo com o estado do cenário (Ex. Número de camiões, quantidade e variedade das uvas). O problema é complexo devido ao elevado número de restrições e variáveis, mas também devido às decisões feitas em cada adega que não levam necessariamente à decisão ótima, porque como é analisado na Secção 4.3 certas decisões menos prováveis podem levar a melhores decisões.

Considerando o seguinte cenário em que dois tipos de uvas: G_1, que se deterioram muito rapidamente, e G_2, que se deterioram lentamente. A adega W tem 2 prensas P_1 e P_2 e um único tegão GT, que está parado em t_i. P_1 está vazio e P_2 está a 75% de capacidade, contendo uvas do tipo G_1. Em t_{i+1}, o camião T_1 chega contendo uvas da variedade G_2, que não pode ser misturada com a variedade G_1. À primeira vista, a solução ideal passaria por ter o GT a começar a carregar as uvas do camião T_1 para a prensa P_1. No entanto, ao considerar a chegada de um novo camião T_2 em t_{i+2}, contendo uvas da variedade G_1. Agora o camião terá que esperar por um período mais longo de tempo, visto que o único tegão disponível está ocupado a descarregar o camião T_2. Consequentemente, as uvas do T_2 estão a degradar-se e a prensa P_2 está parada à espera de estar na sua capacidade máxima antes de começar o processamento das suas uvas. Neste cenário, a melhor solução seria fazer com que o tegão esperasse até t_{i+1} e ser usado em t_{i+2} para descarregar o camião T_2, que contem as uvas mais frágeis. Isto iria não só permitir que as uvas fossem mais rapidamente descarregadas, portanto também evitar a sua degradação, como também iniciar o processamento da prensa P_2 mais rápido.

Garantidamente, e como previamente discutido, não é possível para a adega prever quais uvas serão entregues a que momento. No entanto, este hipotético e simples exemplo é dado para transmitir a noção que neste problema, a otimizações locais não levam necessariamente à solução global ótima, e que gerir os camiões usando *first-in-first-out* (FIFO), também não garante uma solução ótima.

3.1 Arquitetura

Para este projeto foi desenvolvida a arquitetura apresentada na Figura 13, esta é uma arquitetura genérica que é para utilizada para a configuração do GA, simulação do cenário, obtenção e análise dos resultados do mesmo por parte do utilizador. Este será desenvolvido em duas fases, a primeira que é o foco deste projeto, em que será implementado o essencial para a configuração, simulação e obtenção da solução mais apta, assim como uma interface para a análise dos resultados por parte do utilizador. A segunda fase irá focar na otimização do tempo que os utilizadores terão que dispensar para a configuração do GA, nela será desenvolvido um Módulo de integração com ERPs e a interface do utilizador será aprimorada.

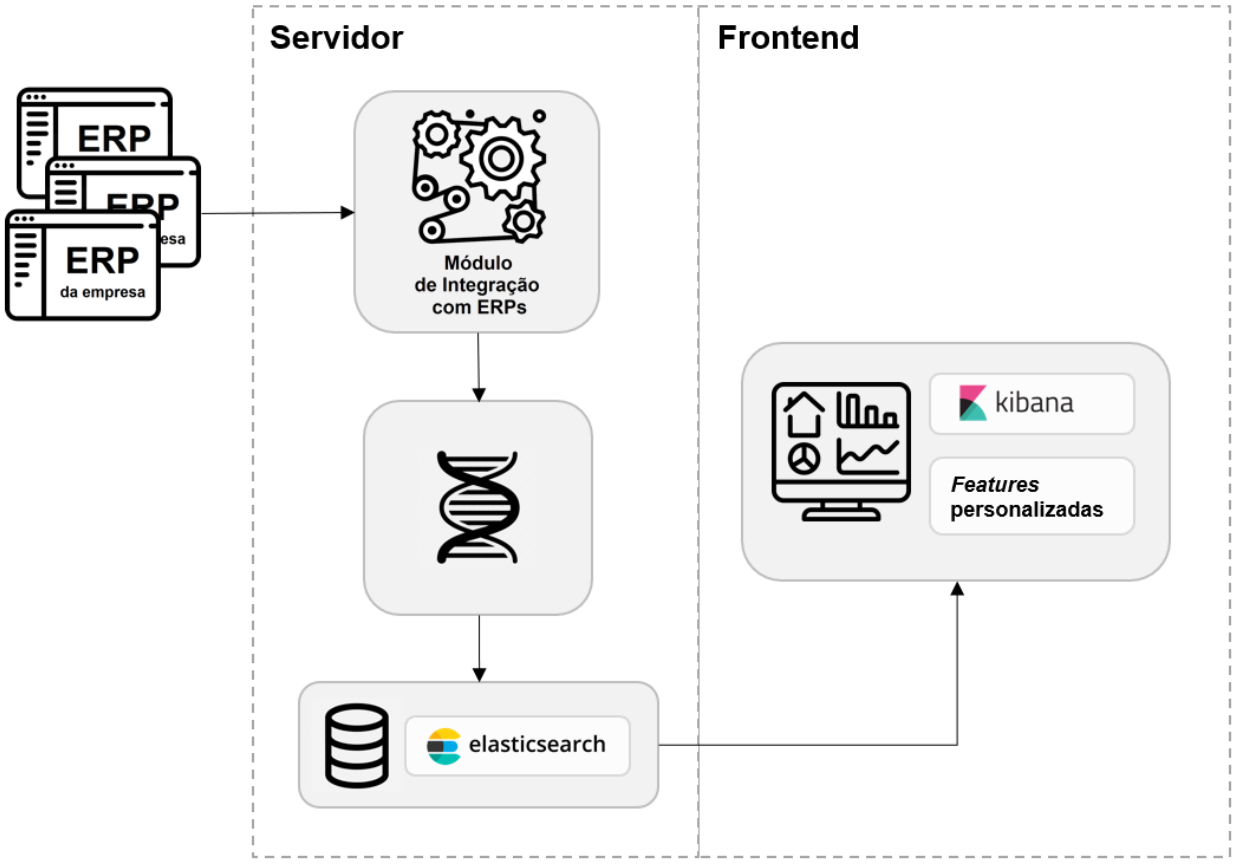


Figura 13: Arquitetura do sistema desenvolvido

A primeira fase, como referido anteriormente, é o desenvolvimento da arquitetura base, ou seja, o essencial para a implementação do GA e análise dos dados gerados pelo mesmo. Com esta, o armazenamento dos dados gerados será feito no elasticsearch e a plataforma kibana é usada para a implementação de dashboards, que são usados para visualização da informação sobre a solução

mais apta encontrada pelo GA em cada simulação efetuada.

Quanto à segunda fase, irá permitir que o utilizador não necessite de efetuar a configurção do GA a cada simulação, visto que será feita a integração com o sistema de ERP destas. Cada empresa possui um sistema de ERP onde são guardadas todas as informações acerca da mesma, estando entre estas, as informações necessárias para a configuração do GA, tais como número de prensas que possui, número de tegões, entradas de camiões e suas informações, etc. No entanto, de empresa para empresa este normalmente é diferente, portanto, e com o objetivo de minimizar o tempo que um utilizador gasta a configurar uma nova realidade do seu cenário, com esta integração o utilizador não irá necessitar de o fazer. Sempre que exista uma mudança nas variáveis do cenário, o módulo de integração com o ERP é informado dessa mudança e irá desencadear uma nova simulação no GA com o novo cenário automaticamente configurado. Deste modo assim que um camião entra na empresa e o sistema de ERP é informado, automáticamente o GA começa a sua busca pela solução mais apta para a resolução deste novo cenário, tirando da equação o tempo que seria necessário para a sua configuração e descartando possíveis erros humanos.

De modo a validar a primeira fase desta arquitetura, esta foi utilizada em três casos de estudo concretos, estando estes descritos na Secção 4.

3.2 Modelação de uma solução

Para modelar e atacar este problema, neste artigo foi usado um GA, para o qual o componente chave é o cromossoma, que representa uma solução para o problema. A solução é composta por um grupo de parâmetros, também chamados de genes, que são as características de uma solução. Neste trabalho, a solução é modelada como um vetor V. Cada posição ($V_1, 0 <= i < n$) representa um minuto sequencial da operação da adega (de tamanho n). O índice i do vetor representa portanto um momento específico no tempo. Nesta posição, o vetor pode conter uma decisão, que representa o momento em que a adega começa uma nova ação (ex. Decidir que um camião irá descarregar num determinado tegão). A maioria das posições do vetor V estão livres, sendo que estas representam momentos em que nenhuma decisão é tomada. Decisões não são imediatas, isto é, cada uma tem a duração que dependendo do estado do mundo no momento e das características das máquinas que estão a ser usadas.

Existem três tipos de decisões: 1) descarregar as uvas de um camião (para um tegão); 2) enviar as uvas do tegão para a prensa; e 3) esmagar as uvas (usando uma prensa). Cada uma destas decisões tem um impacto no estado do mundo que se estende ao longo do tempo. Isto é, quando um camião é atribuído a um tegão, ele irá descarregar as uvas até que o camião fique vazio, que

depende de várias variáveis.

O primeiro tipo de decisão é representado por um tuplo de 2 elementos $U = (T, GT)$ quando T representa o camião do qual as uvas estão a ser descarregadas e GT representa o tegão que está a ser usado.

O camião é representado por um tuplo de 4 elementos $T = (id_gt, G, l, of)$, dos quais os elementos representam, o identificador único do camião, a variedade de uvas transportada pelo camião, a carga atual do camião e a velocidade de descarga (em Kg/min). Cada variedade de uvas é também representada por um tuplo de 2 elementos $U = (id_g, f)$, no qual id_g representa a variedade da uva (ex. Loureiro, Fernão Pires), e f representa a fragilidade da uva (ex. O quão rápida ele se deteriora ao longo do tempo).

O tegão é representado por um tuplo de 5 elementos $GT = (id_gt, G, l, of)$ cujos elementos representam, o identificador único do tegão, a variedade de uva que ele contém (se tiver), à carga atual, a capacidade máxima do tegão, e a sua velocidade de descarga (em Kg/min).

O segundo tipo de decisão é representado por um tuplo de 3 elementos $S = (GT, G, PM)$ onde GT representa o tegão, G a variedade das uvas que estão atualmente no tegão, e PM que representa a prensa para onde as uvas serão enviadas.

A prensa é representada por um tuplo de 5 elementos $PM = (id_pm, G, l, s)$ cujos elementos são, o identificador da prensa, o tipo de uvas que está atualmente na prensa (se existir algum), a atual carga da prensa, a capacidade máxima e o seu estado atual (a processar ou não), respetivamente.

Finalmente o terceiro tipo de decisão (iniciar o ciclo da prensa) é representada simplesmente pelo *singleton* $P = (PM)$.

3.3 Inicialização

No início do GA, um grupo inicial de soluções é criado. Isto é chamado de população inicial. Neste projeto foi usado um *fixed-size population model* que significa que o número de soluções N em cada geração do GA irá permanecer a mesma ao longo da execução do algoritmo, visto que o algoritmo não deve ter uma população demasiado grande, porque esta afeta a performance do mesmo, nem uma população demasiado baixa, porque reduz drasticamente o seu espaço de pesquisa. Cada solução na população inicial é criada aleatoriamente, na tentativa de cobrir todo o espaço da pesquisa o mais possível. A criação de um número aleatório requer os seguintes *inputs*:

- n - o tamanho do período de tempo a ser otimizado, que se traduz para o tamanho do vetor V

- pr_d - a probabilidade de a cada momento criar uma nova decisão aleatória ([0..1])
- pr_u - a probabilidade de gerar a decisão de descarregar as uvas, sempre que uma nova decisão é gerada ([0..1])
- pr_{gt} - a probabilidade de gerar a decisão de enviar as uvas de um tegão para a prensa, quando uma decisão é gerada ([0..1])
- pr_p - a probabilidade de gerar a decisão de iniciar o processamento de uma prensa, quando uma nova decisão é gerada ([0..1])
- Tl - a lista de camiões em espera para descarregar as suas uvas
- Gl - a lista de tegões
- Pl - a lista de prensas

, para o qual pr_u + pr_{gt} + pr_p = 1

Uma solução aleatória é criada da seguinte maneira. Primeiro, um vetor do tamanho n é inicializado. De seguida o vetor é percorrido. Por cada posição i entre 0 e n-1, uma nova decisão é criada com a probabilidade pr_d. A decisão será uma das três possíveis, de acordo com as respetivas probabilidades pr_u, pr_{gt} e pr_p.

Sempre que uma nova decisão é gerada, os seus elementos são selecionados aleatoriamente da lista de *inputs*. Portanto, se a decisão de descarregar as uvas é selecionada, um camião de Tl e um tegão de Gl são selecionados aleatoriamente. Quando a decisão de processar as uvas for gerada, uma prensa é selecionada aleatoriamente.

Como dito anteriormente, esta inicialização aleatória é feita para cobrir o máximo possível do espaço de pesquisa. No entanto, frequentemente acontece que uma dada solução pode não ser válida no sentido de conter decisões que, nesse momento em particular, não são possíveis. Por exemplo, pode haver uma decisão de descarregar um camião que no momento já se encontra vazio, ou uma decisão de começar o processamento das uvas na prensa que já se encontra a processar nesse momento. Enquanto que a existência de soluções não admissíveis podia ser evitada usando heurísticas, elas são permitidas porque facilitam a inicialização do algoritmo, assim como na exploração do espaço de pesquisa. Além disso, uma solução não admissível pode tornar-se admissível com a aplicação de operadores genéticos, como mutação e crossover.

Apesar disso, quando é calculado o fitness de cada solução ou quando são apresentadas ao utilizador, apenas as decisões válidas são consideradas numa solução.

O processo de inicialização do GA desenvolvido neste trabalho é representado no Algoritmo 1.

Output: Uma lista S de soluções inicializadas
function PopInit($N, n, pr_d, pr_u, pr_{gt}, pr_p, Tl, Gl, Pl$)
    $S \leftarrow \emptyset$;
    $i \leftarrow 0$;
    while $i < N$ **do**
        Inicializar o vetor v de tamanho n;
        $j \leftarrow 0$;
        while $j < n$ **do**
            $r_1 < random([0..1])$;
            if $r_1 < pr_d$ **then**
                $r_2 \leftarrow random([0..1])$
                if $r_2 < pr_u$ **then**
                    $tr \leftarrow selectRandomFrom(Tl)$;
                    $gt \leftarrow selectRandomFrom(Gl)$;
                    $v[j] \leftarrow U(tr, gt)$;
                else
                    if $r_2 < pr_u + pr_{gt}$ **then**
                        $pr \leftarrow selectRandomFrom(Pl)$;
                        $gt \leftarrow selectRandomFrom(Gl)$;
                        $v[j] \leftarrow U(gt, pr)$;
                    else
                        $pr \leftarrow selectRandomFrom(Pl)$;
                        $v[j] \leftarrow P(pr)$;
                    end
                end
            else
                $v[j] \leftarrow null$;
            end
        end
        Adiciona v a S;
    end
    Retorna S;
end

Algorithm 1: Otimização do processo de receção de uvas

A Figura 14 representa visualmente uma simplificação de uma possível solução para um problema hipotético, assim como a evolução do estado do mundo quando é aplicado. O exemplo

retratado na Figura não é realista, meramente tem o propósito de representar a estrutura de uma solução e não como são usadas. Para este exemplo, uma configuração simples é considerada que contém 2 camiões (cada um carrega tipos de uva diferentes), um tegão com a capacidade de 10 toneladas, e 2 prensas também com a capacidade de 10 toneladas.

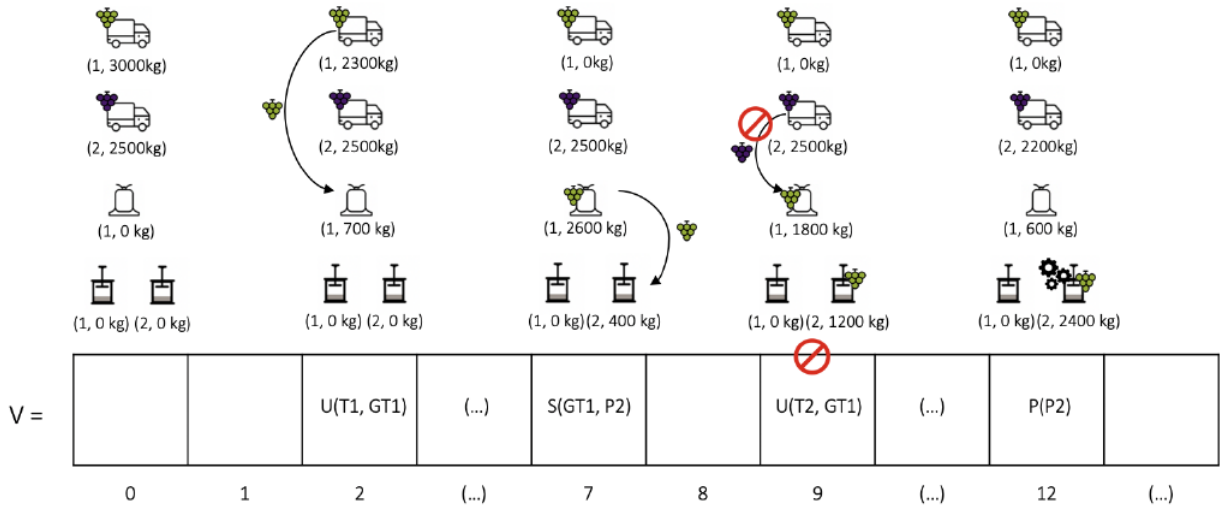


Figura 14: Um exemplo de como seria uma solução gerada aleatoriamente e o seu efeito no estado do mundo enquanto é executada

Considerando que ambos os camiões descarregam as suas uvas a uma taxa de 700Kg/min, que o tegão processa as uvas a uma taxa de 400Kg/min e que um ciclo da prensa demora 10 minutos. O vetor V retrata uma possível solução gerada aleatoriamente para este problema. A primeira decisão foi gerada em $t = 2$ e consiste na descarga das uvas do camião 1 para o tegão. Como o tegão está disponível e existem uvas no camião 1, a ação começa. Dada a taxa de descarga do camião, esta ação acaba em $t = 6$, o momento em que o camião está vazio. Em $t = 7$ uma nova decisão começa a ser executada, que consiste em enviar as uvas do tegão para a segunda prensa. Sabendo que: 1) a prensa não está a processar uvas neste momento; 2) as uvas são compatíveis ou a prensa está vazia; e 3) a prensa tem capacidade suficiente, portanto a ação começa. Dada a taxa de descarga do tegão, esta operação vai até $t = 14$, se não for interrompida.

Em $t = 9$, existe a decisão de começar a descarregar as uvas do segundo camião para o tegão. Contudo, e como o tegão tem capacidade disponível, as uvas de variedades diferentes não podem ser misturadas. Portanto esta decisão não é executada. Em $t = 12$ existe a decisão da segunda prensa começar o seu ciclo. Como a prensa tem uvas, a ação começa. Isto força o tegão a parar de lhe enviar uvas.

A solução parcial desta solução, demonstrada na Figura 14 é meramente um exemplo do que é uma solução gerada aleatoriamente e como poderia potencialmente ser: uma sequência de decisões que claramente não estão otimizadas e que poderá incluir decisões inválidas dado o estado do mundo no momento. Potenciais otimizações a esta solução poderiam incluir:

- Mover a primeira decisão para $t = 0$ de modo a que o processo comece mais cedo e otimize o tempo de espera das uvas;
- Mover a decisão de $t = 7$ para $t = 1$ de modo que o tegão comece a enviar as uvas para a prensa assim que o camião comece a descarregar, permitindo estas duas ações serem executadas simultaneamente. Isto iria permitir que o tegão ficasse vazio em $t = 8$;
- Mover a decisão de $t = 12$ para $t = 9$. Isto iria fazer com que não só começasse mais cedo, como também garantir que a prensa já teria recebido todas as uvas do tegão, portanto libertava o tegão para começar a receber as uvas do segundo camião e enviá-las para a outra prensa.

Estas otimizações são um exemplo do que é feito pelo Algoritmo Genético durante a fase de reprodução. Contudo, enquanto que no exemplo apenas foi considerado mover decisões, o algoritmo considera também a remoção de decisões existentes assim como a criação de novas. Completando estas pequenas mudanças aleatórias e selecionando as melhores soluções em cada geração, o algoritmo move para a população as melhores soluções.

3.4 Função de fitness e estratégia de seleção

Tendo formalizado a natureza da solução, outro elemento fundamental nos GA é a habilidade de medir a qualidade. O quão boa é uma solução é dado pela função de fitness. O objetivo desta função é qualificar, de uma forma objetiva e comparativa, o quão boa é uma solução. Contudo, o seu valor não é absoluto no sentido de que é impossível saber se a melhor solução possível foi encontrada. A função de fitness permite apenas comparar um solução com a próxima.

No contexto deste trabalho, a função de fitness considera 3 comportamentos principais:

- O tempo que as uvas gastam em cada fase do processo (ex. no camião, no tegão, na prensa antes do seu processamento iniciar), assim como a sua quantidade e variedade;
- O tempo que o condutor gasta à espera da descarga;
- O tempo perdido até que o processo termine;

Idealmente, a adega tem o objetivo de minimizar estes três componentes. Contudo, nem sempre é possível. Frequentemente, para otimizar o tempo que cada variedade de uvas espera (ex. minimizando o tempo de espera das uvas mais frágeis), os outros dois componentes são prejudicados (ex. aumentando a média do tempo de espera, ou o tempo de espera de alguns condutores).

Portanto, a função de fitness pode ser ajustada de acordo com os objetivos da adega afinando três pesos, $(w_g, w_d$ e $w_e)$ que representam, a importância relativa dos três componentes, sendo que a sua soma é igual a 1. Por exemplo, se a adega quiser focar na satisfação dos condutores o maior peso deve ser atribuído ao segundo componente, que deve gerar soluções mais próximas da distribuição FIFO. Por outro lado, se a adega quiser focar na qualidade do vinho e no processamento de uvas de acordo com a sua fragilidade, os tempos de espera podem variar significativamente entre condutores, e a sua satisfação poderá ser mais baixa.

A função de fitness é também afetada pela fragilidade das uvas (que é definida pelo utilizador na configuração do sistema) e por outros três pesos w_1, w_2, w_3 que representam o custo relativo das uvas que aguardam em cada fase do processo. É esperado que haja um menor impacto nas uvas quando estas esperam na prensa em vez de no camião, visto que elas se encontram mais perto do fim do processo. A diferença nestes pesos dá preferência a soluções cujas uvas estejam mais avançadas no processo.

De modo a calcular o fitness de uma dada solução, esta é primeiramente simulada. Isto é, dado um estado inicial do mundo, as decisões são aplicadas e afetam a simulação. Isto é feito percorrendo o vetor de decisões e atualizando o estado do mundo de acordo. Enquanto isto está a ser feito, as seguintes variáveis são calculadas:

- A quantidade de cada variedade de uvas à espera em cada fase do processo (ex. camião, tegão, prensa) a qualquer momento. Estas são depois multiplicadas pela fragilidade correspondente e por w_1, w_2 e w_3 de acordo com o respetivo estado (ex. camião, tegão, prensa);
- Quanto tempo cada condutor espera até que todas as uvas que carrega sejam descarregadas;
- Quanto tempo demora até que todas a uvas sejam processadas.

No fim, a soma dos pesos destes três componentes é calculada, de acordo com os pesos w_g, w_g e w_g, que resulta no valor do fitness. O valor do fitness é calculado para cada solução da população, em cada geração, e é usado como base para selecionar os melhores indivíduos de cada geração. Portanto, as soluções são ordenadas de acordo com o seu valor de fitness e os melhores indivíduos são selecionados para serem reproduzidos e gerar a próxima geração.

3.5 Reprodução e seleção

Em GA, existe também a necessidade de utilizar operadores genéticos que introduzem pequenas variações nas soluções existentes, para que novas e potencialmente melhores possam ser encontradas, que melhoram de forma geral o fitness ao longo das sucessivas gerações. Neste projeto foram considerados 2 operadores padrão: mutação e crossover. Ambos os operadores mudam as decisões do vetor.

Mutação é um operador unário que tem como *input* inicial uma solução e produz uma nova com pequenas mudanças feitas na mesma. Foram consideradas quatro tipos de mudanças:

- Apagar uma decisão aleatoriamente escolhida;
- Mover uma decisão escolhida aleatoriamente para um posição próxima que esteja livre. O alcance das posições candidatas é determinado pela variavel mf, que indica o fator de mutação;
- Adicionar uma nova decisão aleatória, a uma posição livre aleatória do vetor;
- Trocar 2 decisões aleatoriamente;

Crossover, por outro lado, é um operador binário: Ele tem como *input* 2 soluções e produz 2 novas soluções. Neste foi considerado realizar crossover num único ponto. Portanto, uma posição do vetor de decisões é escolhida aleatoriamente, sendo usada para separar o vetor de ambos os pais em 2. As duas novas soluções são obtidas através da junção da primeira parte da primeira solução com a segunda parte da segunda solução, e vice versa.

Estes 2 operadores de reprodução são aplicados a cada uma das melhores soluções de cada geração, com igual probabilidade. Após a reprodução, uma nova geração é criada que contém os melhores elementos da anterior, assim como os novos criados através da mutação e crossover.

O processo volta à avaliação do fitness de cada solução e selecionando as melhores. Isto repete-se até que um ou mais dos seguintes critérios de paragem sejam atingidos:

- Um número pré-definido de gerações é atingido;
- Passa um certo período de tempo;
- O fitness das melhores soluções não muda por mais de δ ao longo de n gerações consecutivas (convergência);

3.6 Aplicação do cliente

Para que haja a interação com o utilizador final com o GA, uma aplicação web é necessária. Um protótipo da mesma foi desenvolvido e pode ser visualizado nas Figura 15 e 16.

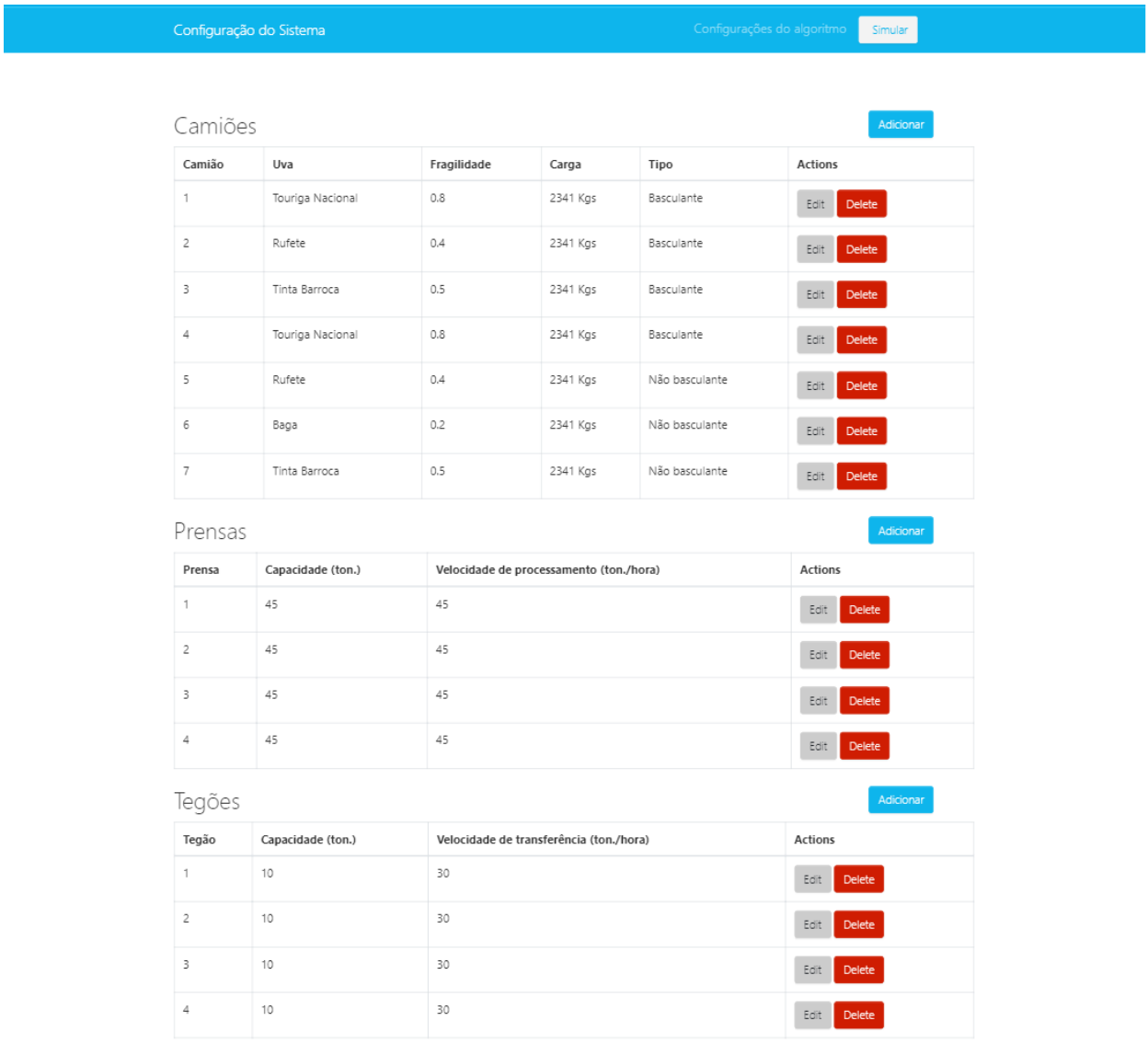


Figura 15: Protótipo da página web para a configuração das entidades do GA

Nesta página os colaboradores responsáveis pela receção dos camiões podem definir todas as variáveis necessárias para a execução do algoritmo para um dado cenário. Este possui três tabelas, para a configuração dos camiões, prensas e tegões. Ao adicionar uma nova entidade a cada uma das tabelas, uma modal irá aparecer e o colaborador poderá preencher todos os dados relativos

à entidade selecionada, como pode ser visualizado na Figura 17. Após todas as entidades serem configuradas, o colaborador pode iniciar a simulação do cenário através do botão superior direito "simular", como pode ser visualizado na Figura 15.

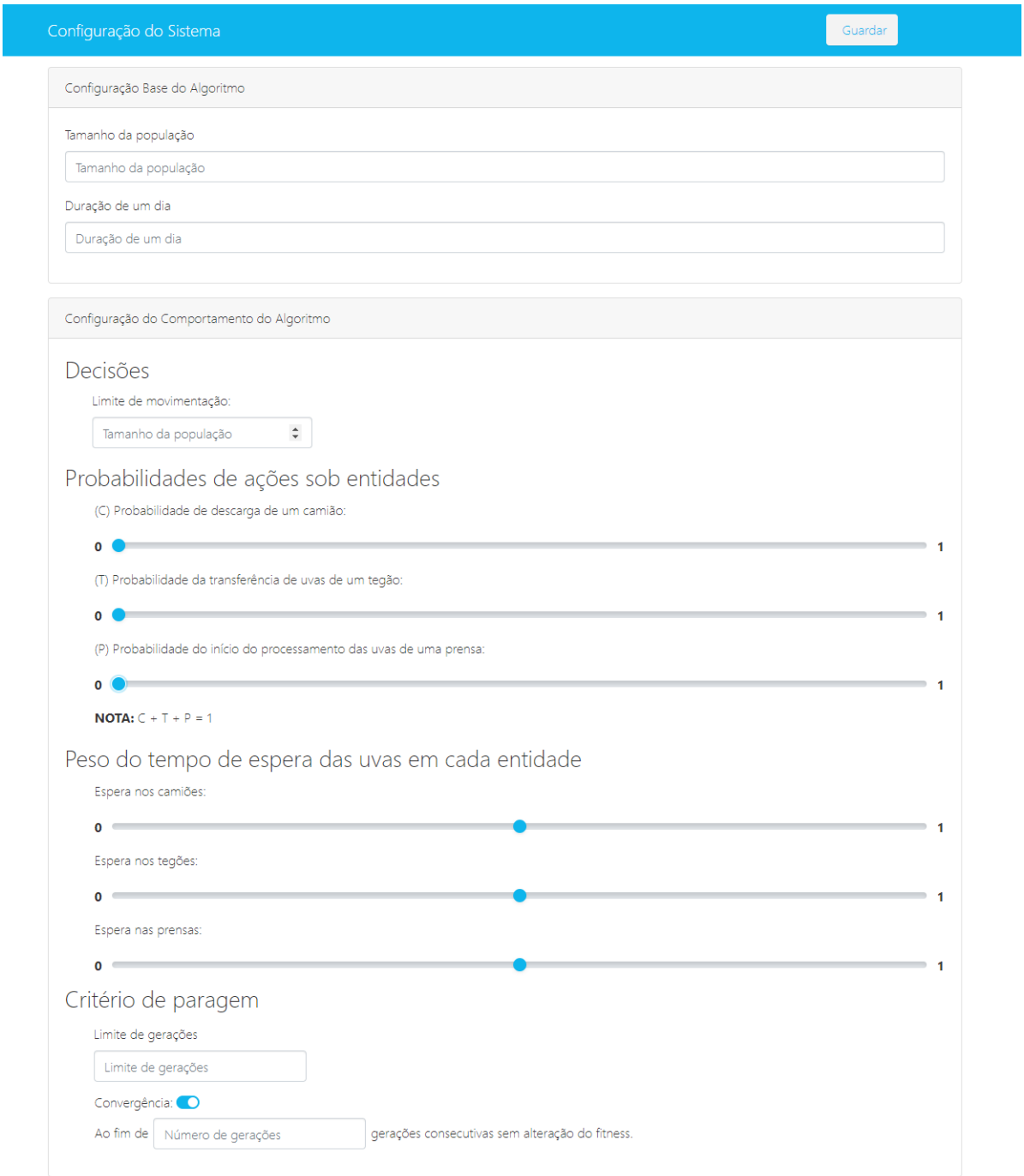


Figura 16: Protótipo da página web para a configuração do GA

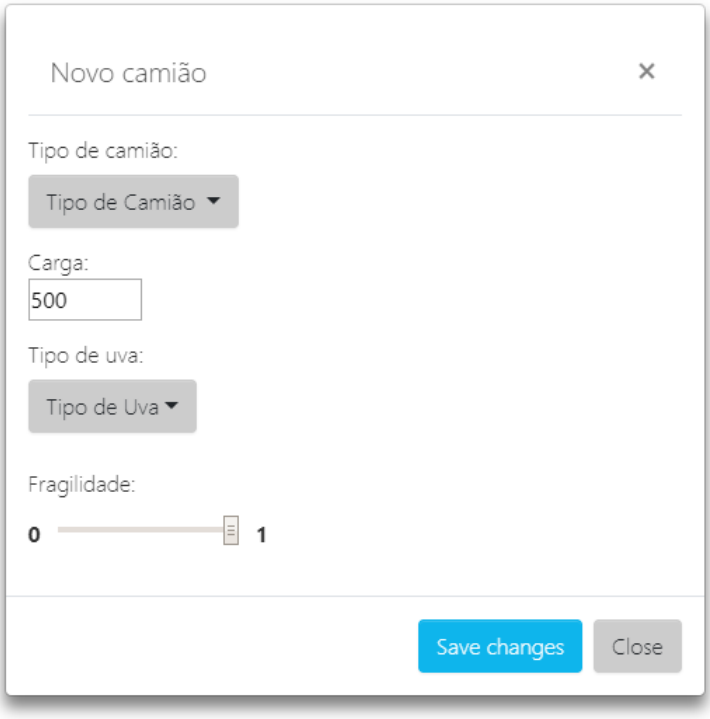


Figura 17: Modal de adição de um camião ao cenário

Assim que a simulação iniciar, uma modal irá aparecer, dando a informação do tempo estimado ao colaborador, assim como quando tempo já passou desde o inicio da mesma. Este pode também cancelar a simulação do cenário através do botão cancelar, sendo que nesta situação a aplicação mostra os resultados da melhor solução encontrada até ao momento. A modal pode ser visualizada na Figura 18.

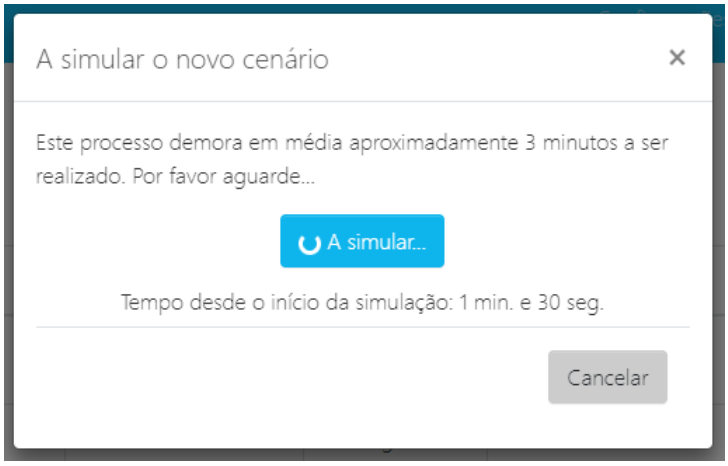


Figura 18: Modal de adição de um camião ao cenário

No que diz respeito à configuração do GA, esta pode ser feita numa página destinada para o efeito, que pode ser visualizada na figura 16. Ela está dividida em duas secções, ”Configuração base do Algoritmo” e ”Configuração do comportamento do algoritmo”. A primeira possui dois *inputs*:

- **Tamanho da população**, que representa o número de indivíduos que uma população deve possuir;
- **Duração de um dia**, que representa o número de minutos de trabalho de um dia, sendo que cada minuto representa uma decisão.

A segunda secção possui as seguintes variáveis:

- **Limite de movimentação**, que representa o número de decisões que uma decisão pode avançar ou recuar no vetor de decisões. Este evento ocorre durante a mutação;
- **Probabilidade de descarga de um camião**, que representa a probabilidade de gerar a decisão de descarregar as uvas, sempre que uma nova decisão é gerada;
- **Probabilidade da transferência de uvas de um tegão**, que representa a probabilidade de gerar a decisão de enviar as uvas de um tegão para a prensa, quando uma decisão é gerada;
- **Probabilidade do início do processamento das uvas de uma prensa**, que representa a probabilidade de gerar a decisão de iniciar o processamento de uma prensa, quando uma nova decisão é gerada;
- **Espera nos camiões**, que representa o quão prejudicial é as uvas esperarem nos camiões, sendo 1 o valor mais alto da escala.
- **Espera nos tegões**, que representa o quão prejudicial é as uvas esperarem nos tegões, sendo 1 o valor mais alto da escala.
- **Espera nas prensas**, que representa o quão prejudicial é as uvas esperarem nas prensas, sendo 1 o valor mais alto da escala.
- **Limite de gerações**, que é um dos critérios de paragem, este representa o limite de gerações que um GA pode gerar, após atingir este limite o GA termina a execução.
- **Convergência**, através dele o utilizador deve definir se o GA deve parar por ter convergido ou não. Caso a seleção seja positiva, é pedido o limite de gerações em que o fitness não foi alterado.

3.7 Aplicação do servidor

O elasticsearch é um mecanismo *open source* de pesquisa e análise de dados altamente escalável. Este permite guardar, pesquisar e analisar grande volumes de dados quase em tempo real. É geralmente usado como mecanismo/tecnologia base que alimenta aplicações que necessitam de pesquisas complexas. O elasticsearch fornece um sistema distribuído baseado no Apache Lucene (Lucene StandardAnalyzer) para indexação e utiliza uma API REST baseada em JSON. É facilmente configurável, visto que traz já bastantes pré-configurações pré-definidas e esconde bastante complexidade de principiantes. A sua curva de aprendizagem para aprender o básico é curta e com pouco esforço qualquer pessoa consegue se tornar produtiva rapidamente.

Os utilizadores começaram a usá-lo para armazenar logs, mas era também necessário conseguir visualizá-los. Para isso a Elastic possui o Kibana, este permite visualizar os dados previamente armazenados no elasticsearch. É possível, através dele, escolher o modo como os dados serão mostrados, podendo ser selecionados desde os gráficos clássicos (histogramas, gráficos de linhas, circulares e assim por diante) até desenhar as próprias visualizações ou até adicionar informação geográfica num mapa. É possível também efetuar análises temporais avançadas, encontrar relações entre os dados visualmente e explorar anomalias através de machine learning.

Todos os dados gerados ao longo do GA desenvolvido foram armazenados no elasticsearch. Deste modo, após a obtenção da solução, todos os dados estarão prontos para a ser utilizados. Foram gerados 3 dashboards:

1. Dashboard que contém toda a informação sobre tempos médios de espera de cada um tipos de uva, assim como a média do fitness ao longo das gerações, representado na Figura 19;

2. Dashboard contendo primeiramente o estado inicial dos camiões, tegões e prensas da adega, de seguida, as decisões que foram efetuadas pela solução mais apta encontrada pelo GA para as três entidades anteriormente descritas, estando representado na Figura 20;

3. Dashboard contendo representações gráficas da carga de cada uma das entidade ao longo do dia da solução mais apta encontrada pelo GA, podendo estes serem visualizados nas figuras 21, 22 e 23.

Começando com o dashboard (1), Figura 19, este é gerado através das informações guardadas no elasticsearch acerca dos tempos médios de espera de cada entidade e média do fitness das soluções geradas ao longo da execução do GA. Este não é um dashboard orientado para o utilizador final, mas foi criado com o intuito de dar uma melhor visão acerca do comportamento e performance do GA. É possível visualizar que na simulação demonstrada na Figura 19 os tempos médios

de espera de cada tipo de uvas vão diminuindo ao longo das iterações do GA, no entanto, visto que as uvas Fernão Pires são as mais frágeis, tendo em conta o exemplo relatado na Secção 4.2, o utilizador consegue ter uma percepção clara de que estas uvas foram preferênciados pelos GA em relação às outras. No gráfico "Tempo médio de espera (Camiões)"é possível visualizar a média do tempo de espera geral dos camiões ao longo das iterações do GA.

Para além dos gráficos acima mencionados foi criado um último que permite ao utilizador verificar o fitness da solução ao longo das iterações do GA, através dele torna-se mais perceptível a evolução do mesmo.

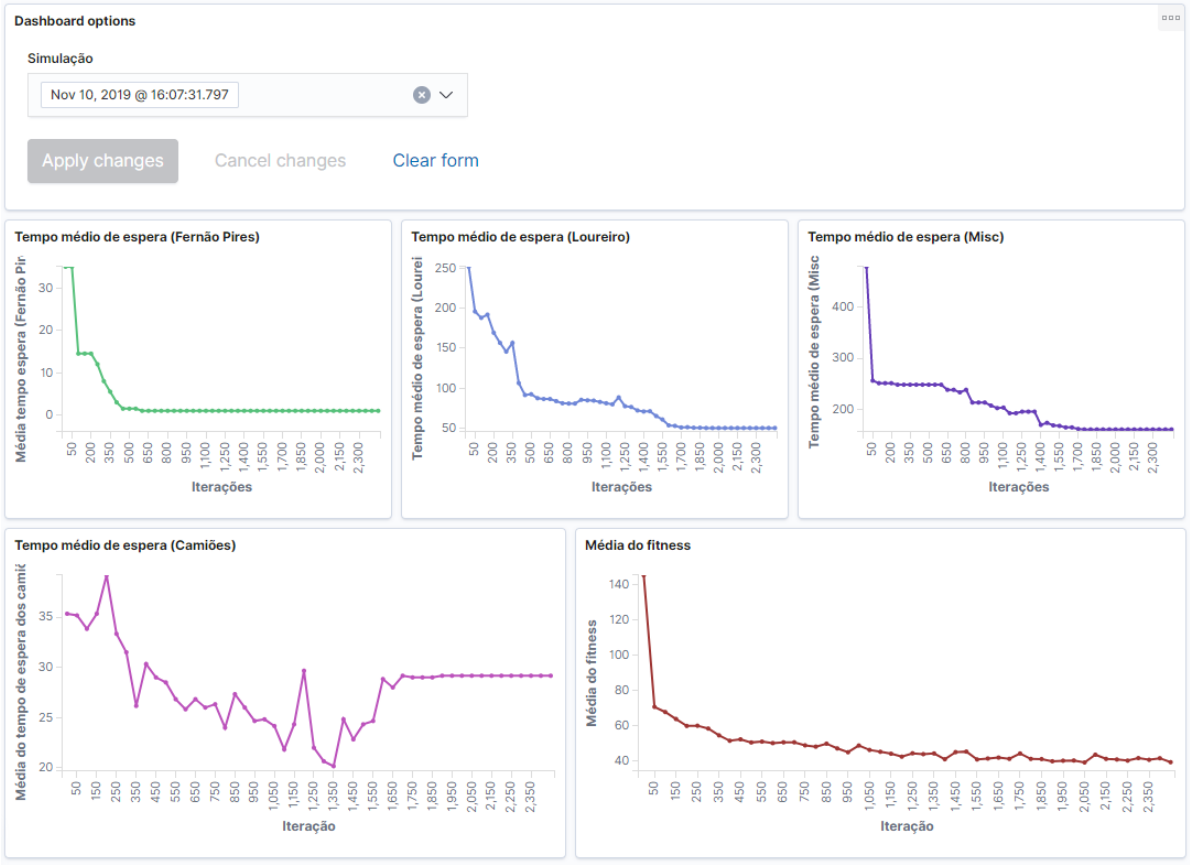


Figura 19: Dashboard dos tempos médios de espera de cada uma das entidades e fitness ao longo das iterações do Algoritmo Genético

O segundo dashboard (2), apresentado na Figura 20, é o dashboard mais importante para o utilizador final. Nele, este conseguirá saber todas as informação sobre cada evento da solução mais apta encontrada pelo Algoritmo Genético. Foram criadas diversas tabelas para demonstrar quais as ações que devem ser realizadas e em que momento, e também o ponto final de cada uma das entidades. As primeiras três tabelas mostram que eventos devem ocorrer em cada uma das

entidades ao longo do tempo. Em todas estas t representa os minutos em que este evento deve ser executado, sendo 0 o ponto inicial. A tabela ”Decisões Camião” indica ao utilizador quando é que cada camião deve enviar as suas uvas para o respetivo tegão. A tabela ”Decisões Tegão” indica ao utilizador quando é que cada tegão deve enviar as suas uvas para a respetiva prensa. Por fim, a tabela ”Decisões Prensa” indica ao utilizador quando é que cada cada prensa deve iniciar o processamento das suas uvas, indicando também qual a carga da prensa após a execução do evento, o tipo de uvas que ficará armazenado na prensa, a sua carga e capacidade máxima. Por último o dashboard termina com as últimas três tabelas a indicarem o estado final de cada uma das entidades após a conclusão de todos os eventos da solução mais apta encontrada pelo GA.

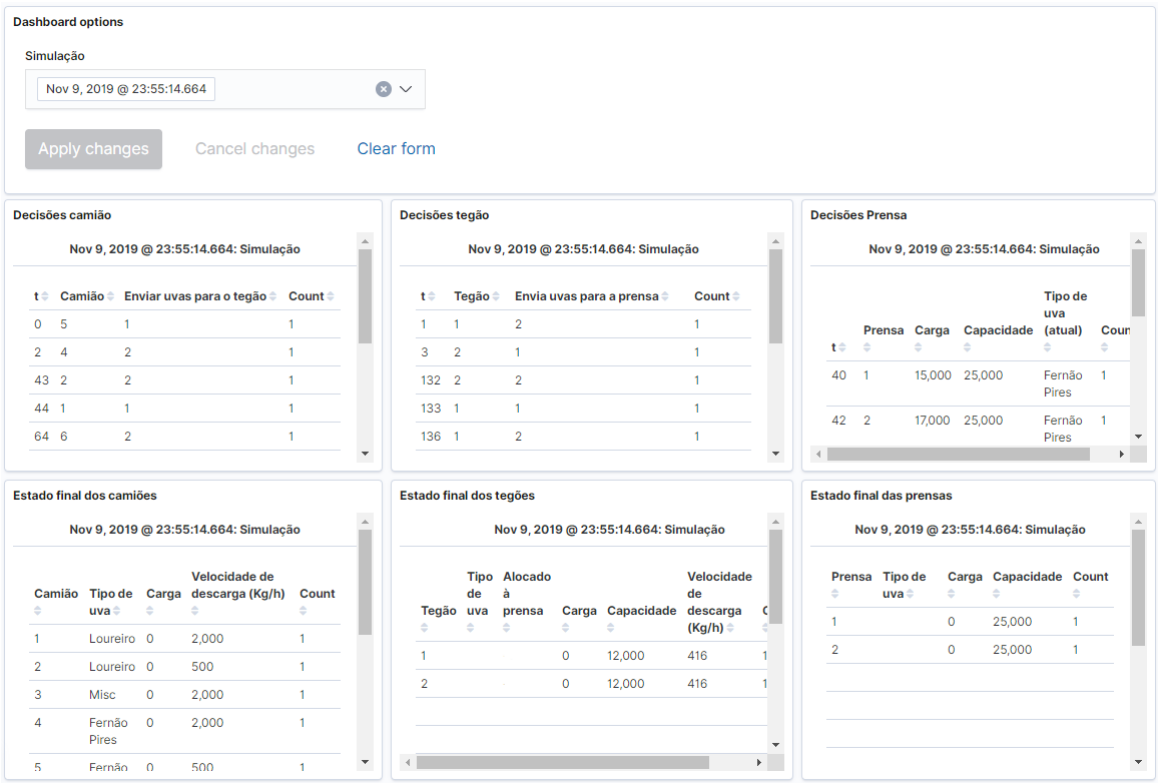


Figura 20: Dashboard da solução mais apta encontrada pelo GA

Quanto ao dashboard (3), que possui os gráficos das figuras 21, 22 e 23, é um dashboard também orientado para o utilizador final, nele este conseguirá ter um percepção visual da carga existente em cada uma das entidades ao longo do tempo. Através da análise destes gráficos o utilizador conseguirá saber de uma forma mais rápida, em comparação com o dashboard anterior que é menos visual, a hora prevista de um certo camião sair da adega, ou quando é que está previsto uma prensa iniciar e terminar o processamento das suas uvas.

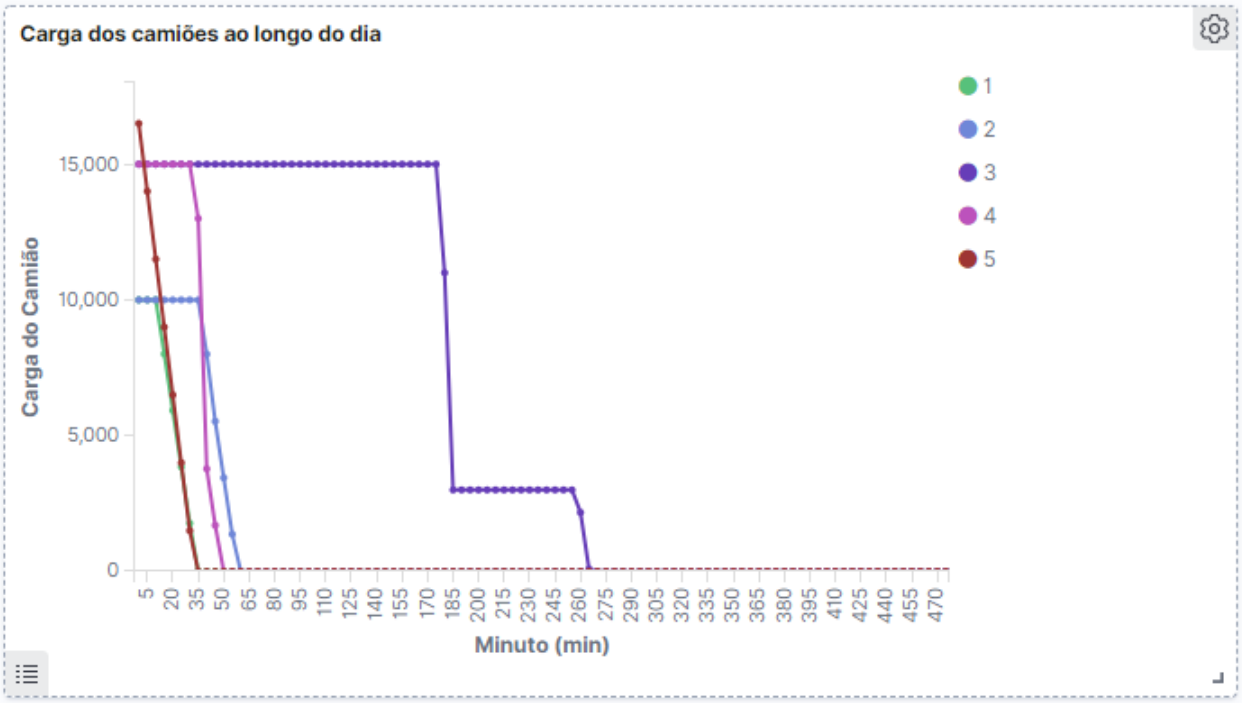


Figura 21: Carga dos camiões ao longo do dia da simulação

Figura 22: Carga dos tegões ao longo do dia da simulação

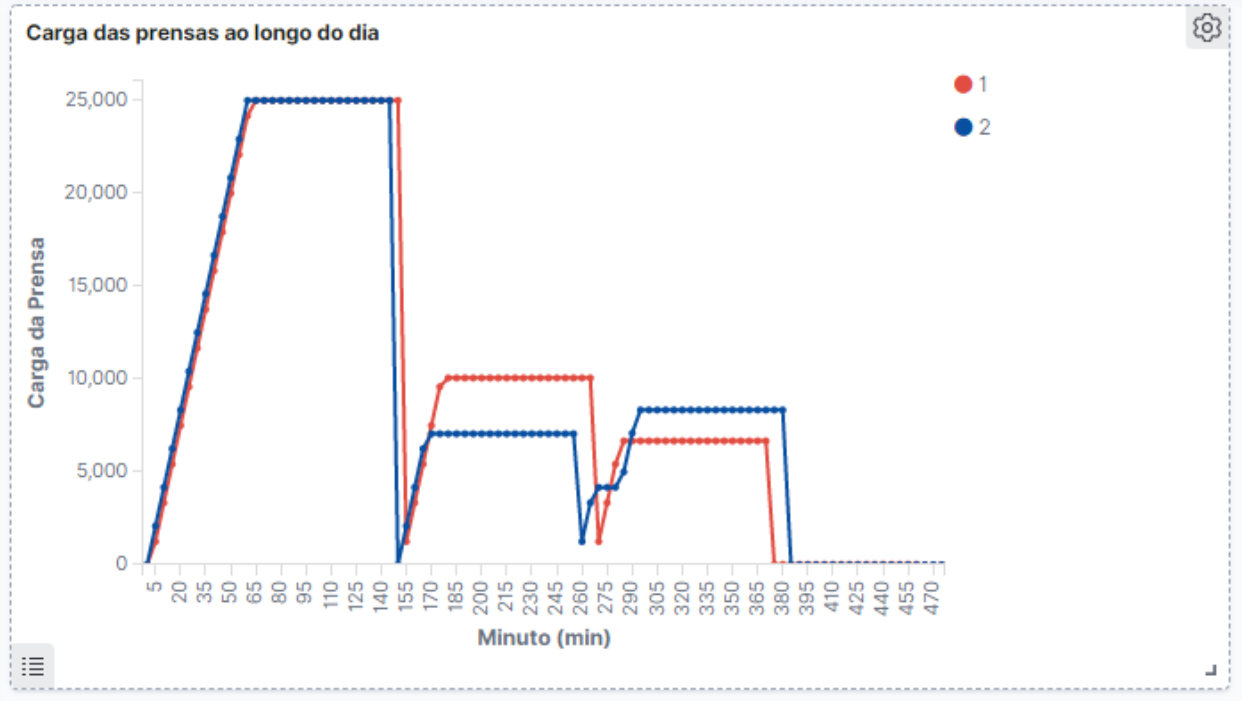


Figura 23: Carga das prensas ao longo do dia da simulação

4 Resultados

De modo a avaliar o sistema desenvolvido, foi considerado o seguinte cenário. A adega tem dois tegões GT_1 e GT_2, com 12 toneladas de capacidade cada. Ambas têm uma velocidade de descarga de 416 kg/min (25 tons/hora). A adega também tem duas prensas P_1 e P_2, cada uma com a capacidade de 25 toneladas e um ciclo de 90 minutos. Considerando a existência de 6 camiões, carregando três variedade diferentes de uvas (que não podem ser misturadas), como detalhado na Tabela 1. A descrição de todas as entidades pode ser visualizada na Figura 24 Como detalhado anteriormente, a fragilidade das uvas é um valor subjetivo atribuído pelo especialista ([0..1]) e indica a fragilidade da uva: quanto maior o valor mais frágil é a uva.

Camião	Uva	Fragilidade	Carga	Tipo
T_1	Loureiro	0.3	10 ton.	basculante (2 ton./min)
T_2	Loureiro	0.3	10 ton.	não basculante (0.5 ton./min)
T_3	Misc	0.2	15 ton.	basculante (2 ton./min)
T_4	Fernão Pires	0.5	15 ton.	basculante (2 ton./min)
T_5	Fernão Pires	0.5	17 ton.	não basculante (0.5 ton./min)
T_6	Fernão Pires	0.3	15 ton.	basculante (2 ton./min)

Tabela 1: Características dos camiões do cenário proposto (valores inventados para serem usados nas abordagens FIFO e GA).

Deve ser notado que as características do problema foram ligeiramente alteradas por uma questão de simplicidade, de modo a tornar mais fácil a documentação. Nomeadamente, um número mais pequeno de camiões foi considerado, apesar da maior quantidade média de uvas para compensar. O ciclo das prensas também é significativamente mais curto (1.5 horas contra as normais 4 horas), para tornar o processo mais curto. Também para facilitar foi assumido que os 6 camiões chegam ao mesmo tempo.

Nesta Secção foram comparadas 2 abordagens para a resolução do problema. Na primeira, o problema é resolvido por um especialista humano que decide baseado na lógica FIFO, seguindo a ordem dos camiões da Tabela 1. A segunda abordagem usa o GA proposto para encontrar a solução para o mesmo problema.

Para avaliar ambas as abordagens, e também para simplificar, a função de fitness considera apenas o primeiro componente, que é a fragilidade das uvas. No entanto, outros fatores são também considerados como os tempos de espera, ou o tempo para concluir o processo.

A comparação entre estas duas abordagens é feita para mostrar que o GA pode igualmente gerar sequências de decisões válidas e lógicas, que frequentemente são melhores que as abordagens

tradicionais.

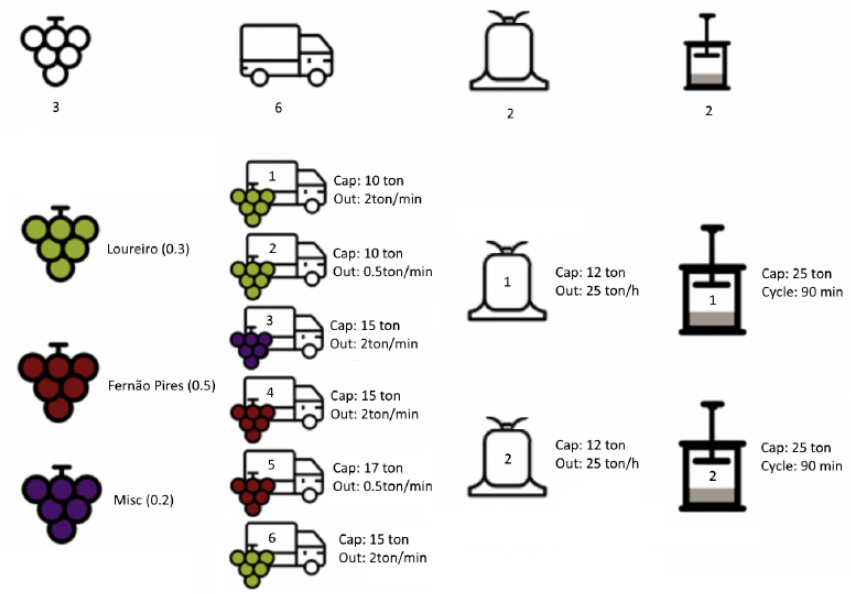


Figura 24: Uma possível configuração do problema para o qual quatro camiões carregados com diferentes tipos de uvas são servidos por dois tegões e duas prensas

4.1 Abordagem FIFO

Esta Secção descreve uma possível estruturação do processo (Tabela 2), como proposto pelo especialista humano. Neste caso, a tradicional abordagem FIFO é usada no qual os camiões são processados pela ordem dada na Tabela 1.

Esta solução começa processando os camiões T_1 e T_2, cada um carregando 10 toneladas de uvas Loureiro. Cada camião é enviado para diferentes tegões, os quais começam imediatamente a enviar as suas uvas para ambas as prensas. Isto significa que em $t = 3$ ambas as prensas P_1 e $P2$ receberam uvas Loureiro. Sabendo que T_1 é um camião basculante, ele acaba de descarregar as uvas em $t = 5$. O T_2, por outro lado, só está vazio em $t = 22$.

Em $t = 26$, GT_1 acaba de enviar as uvas para a P_1 e a prensa começa a processar a uma capacidade de 40%. O T_4 começa a descarregar as uvas no GT_2, que agora se encontra vazio.

Em $t = 33$ e $t = 35$, os GT_1 e GT_2 ficam respetivamente cheios, o que significa que os camiões T_3 e T_4 têm que parar de carregar as suas uvas. Este estado é mantido até $t = 115$, quando a $P1$ acaba o processamento e o G_1 começa a enviar uvas para a P_1. A P_2 acaba em $t = 117$, portanto

t	Decisão	T_1	T_2	T_3	T_4	T_5	T_6	GT_1	GT_2	P_1	P_2
0	$T_1 \Rightarrow GT_1$	8	10	15	15	17	15	2	0	0	0
1	$GT_1 \Rightarrow P_1$	0	10	15	15	17	15	3.6	0	0.4	0
2	$T_2 \Rightarrow GT_2$	4	9.5	15	15	17	15	5.2	0.5	0.8	0
3	$GT_2 \Rightarrow P_2$	2	0	15	15	17	15	6.8	0.4	1.2	0.4
26	P_1 a processar	0	0	15	15	17	15	0	0.1	10	10
27	$T_3 \Rightarrow GT_1$	0	0	13	15	17	15	2	0	10	10
28	P_2 a processar	0	0	11	15	17	15	4	0	10	10
29	$T_4 \Rightarrow GT_2$	0	0	9	13	17	15	6	2	10	10
116	$GT_1 \Rightarrow P_1$	0	0	3	3	17	15	11.6	12	0.4	10
118	$GT_2 \Rightarrow P_2$	0	0	2.2	3	17	15	11.6	11.6	1.2	0.4
128	$T_5 \Rightarrow GT_2$	0	0	0	0	16.5	15	9.6	10.9	5.4	4.6
153	P_1 a processar	0	0	0	0	5.4	15	0	11.6	15	15
154	$T_6 \Rightarrow GT_1$	0	0	0	0	5	13	2	11.6	15	15.4
179	P_2 a processar	0	0	0	0	0	3	12	7	15	25
243	$GT_1 \Rightarrow P_1$	0	0	0	0	0	3	11.6	7	0.4	25
269	$GT_2 \Rightarrow P_2$	0	0	0	0	0	0	3.8	6.6	11.2	0.4
280	P_1 a processar	0	0	0	0	0	0	0	2	15	5
286	P_2 a processar	0	0	0	0	0	0	0	0	15	7
375	Fim	0	0	0	0	0	0	0	0	0	0

Tabela 2: Cronograma da abordagem FIFO. As últimas 10 colunas mostram o carga de cada maquina (em toneladas, arredondado a 1 décima).

o GT_2 pode também começar a enviar-lhe uvas. Os camiões T_3 e T_4 podem portanto continuar a descarga das uvas, acabando em $t = 125$ e $t = 127$, respetivamente.

Quando T_4 deixar o GT_2, o T_5 começa a descarregar no mesmo tegão, visto que ele carrega a mesma variedade de uvas. Entretanto, em $t = 153$, o GT_1 fica vazio e a P_1 começa o seu ciclo a 60% de capacidade. O último camião, T_6, pode portanto ser atribuído a um tegão vazio. Ele começa a descarregar as uvas, mas pára em $t = 160$, porque o tegão fica cheiro enquanto a P_1 se encontra a processar. O T_5 acabada a descarga das uvas em $t = 168$ e vai embora. Em $t = 179$, a P_2 chega à sua capacidade máxima e começa o processamento das uvas.

A prensa P_1 acaba o seu ciclo em $t = 243$ e o GT_1 começa a enviar as suas uvas. O T_6 vai embora em $t = 252$ e a P_2 acaba em $t = 268$, começando o GT_2 a enviar as uvas. O GT_1 e GT_2 acabam de enviar as uvas para as respetivas prensas em $t = 280$ e $t = 286$, respetivamente, e ambas as prensas começam os seus ciclos. O processo acaba em $t = 375$, com o fim do ciclo de P_2.

No que diz respeito à avaliação desta possível solução, os condutores esperaram um total de

353 minutos, ou uma média de 58,83 minutos por condutor, para começar a descarregar as uvas. Quando são considerados estes resultados por tipo de uva carregada, os condutores que carregam as uvas Fernão Pires, consideradas as mais frágeis, esperaram uma média de 78.5 minutos enquanto que condutores que carregavam uvas do tipo "Loureiro" e "Misc" esperaram, em média, 52 e 27 minutos, respetivamente. Esta solução requer um total de 18 decisões e 6 ciclos de prensas (3 cada). A média de capacidade das prensas quando iniciavam o seu ciclo é de 54,67%, bem abaixo do ótimo. O fitness desta solução, quando calculado com a solução fitness proposta é de 58,83.

4.2 Abordagem GA

Nesta Secção é detalhada uma solução descoberta pelo GA proposto, para o mesmo cenário. O GA foi configurado da seguinte maneira. Uma população de 50 indivíduos foi usada, com uma duração de 480 minutos cada. Em cada iteração as melhores 25 são selecionadas para serem reproduzidas. Cada uma das soluções selecionadas tem a mesma probabilidade de mutar ou ser cruzada com outra solução aleatória. Cada nova geração é gerada através das 25 melhores da geração anterior mais as 25 novas soluções geradas através dos operadores de reprodução. O algoritmo pára depois de 2500 iterações.

A solução detalhada na Tabela 3 foi o resultado de uma execução do GA proposto usando uma função de fitness que prioriza a minimização do tempo de espera das uvas de acordo com a sua fragilidade. Os primeiros passos desta solução consistem em atribuir os camiões T_5 e T_4 aos tegões GT_2 e GT_1, respetivamente. Isto permite processar assim que possível os dois tipos de uvas que contêm as uvas mais frágeis. O T_4 vai embora em $t = 12$ e em $t = 23$ a prensa P_2 começa o seu ciclo, apesar de estar a uma capacidade relativamente baixa no momento (36%) e do facto do GT_2 estar a enviar-lhe uvas. Não obstante, a decisão que pode parecer contra intuitiva à primeira vista, permite à P_2 ficar disponível mais cedo para outras uvas. Além disso, GT_2 é direcionado para a P_1 em $t = 24$. Isto significa que depois deste momento, ambos os tegões estão a enviar uvas para a P_1, que tem agora capacidade para reter todas as uvas restantes da variedade Fernão Pires.

O T_5 deixa a adega em $t = 34$ e o GT_1 acaba de enviar as uvas para a P_1 em $t = 40$. De seguida, e como o GT_1 está agora vazio, o T_2 é atribuído a ele e começa a descarregar as suas uvas Loureiro. Em $t = 43$ o GT_2 também acaba de enviar as suas uvas para a P_1, que começa o processamento a 91% da sua capacidade. De seguida, o T_3 começa a descarregar no GT_2, que atinge a sua capacidade máxima em $t = 50$, fazendo com que o camião T_3 tenha que esperar. O T_2 deixa a adega em $t = 61$ e no momento seguinte o T_6 toma o seu lugar no GT_1. No entanto, GT_1 atinge a sua capacidade máxima e o T_6 tem que esperar.

t	Decisão	T_1	T_2	T_3	T_4	T_5	T_6	GT_1	GT_2	P_1	P_2
0	$T_5 \Rightarrow GT_2$	10	10	15	15	16.5	15	0	0.5	0	0
1	$GT_2 \Rightarrow P_2$	10	10	15	15	16	15	0	0.6	0	0.4
2	$T_4 \Rightarrow GT_1$	10	10	15	13	15.5	15	2	0.7	0	0.8
3	$GT_1 \Rightarrow P_1$	10	10	15	11	15	15	3.6	0.8	0.4	1.2
23	P_2 a processar	10	10	15	0	5	15	6.2	2.8	8.7	9.2
24	$GT_2 \Rightarrow P_1$	10	10	15	0	4.5	15	5.8	2.9	9.6	9.2
41	$T_2 \Rightarrow GT_1$	10	9.5	15	0	0	15	0.5	0.4	22.5	9.2
43	P_1 a processar	10	8.5	15	0	0	15	1.5	0	22.8	9.2
44	$T_3 \Rightarrow GT_2$	10	8	13	0	0	15	2	2	22.8	9.2
62	$T_6 \Rightarrow GT_1$	10	0	3	0	0	13	12	12	22.8	9.2
113	$GT_2 \Rightarrow P_2$	10	0	3	0	0	13	12	11.6	22.8	0.4
133	$GT_1 \Rightarrow P_1$	10	0	0	0	0	13	11.6	6.2	0.4	8.7
150	P_2 a processar	10	0	0	0	0	5.9	11.6	0	7.5	15
151	$T_1 \Rightarrow GT_2$	8	0	0	0	0	5.5	11.6	2	7.9	15
152	$GT_2 \Rightarrow P_1$	6	0	0	0	0	5.1	11.6	3.6	8.7	15
172	P_1 a processar	0	0	0	0	0	0	8.4	1.6	25	15
240	$GT_1 \Rightarrow P_2$	0	0	0	0	0	0	7.9	1.6	25	0.4
241	$GT_2 \Rightarrow P_2$	0	0	0	0	0	0	7.5	1.2	25	1.2
261	P_1 a processar	0	0	0	0	0	0	0	0	25	10
350	Fim	0	0	0	0	0	0	0	0	0	0

Tabela 3: Cronograma de uma solução gerada pelo GA. As últimas 10 colunas mostram o carga de cada maquina (em toneladas, arredondado a 1 décima).

A prensa P_2, que começa mais cedo termina em $t = 112$ e o GT_2 começa a enviar as suas uvas para ela. O T_3 acaba de descarregar em $t = 122$ e a P_1 termina o seu ciclo em $t = 132$. Com a prensa agora livre, o GT_1 começa a enviar-lhe as suas uvas. A P_2, por outro lado, começa o seu ciclo em $t = 150$ (a 60% de capacidade), assim que o GT_2 termina de enviar as uvas. O T_1 é atribuído ao GT_2 em $t = 151$ e o GT_2 começa a enviar as suas uvas para a P_1, em simultaneo com o GT_1. Os camiões T_1 e T_6 deixam a adega em $t = 156$ e $t = 166$, respetivamente.

A P_1 começa o seu ciclo em $t = 172$ na sua capacidade máxima, enquanto que a P_2 acaba em $t = 239$. assim que a prensa fica livre, ambos os tegões começam a enviar-lhe as suas uvas, acabando em $t = 245$ e $t = 261$, respetivamente. Neste momento, a P_2 começa o processamento. O processo conclui com o fim de ambos os ciclos das prensas em $t = 262$ e $t = 350$, respetivamente.

Esta solução gerada automaticamente contém mais uma decisão do que a abordagem FIFO (19 decisões). No entanto, conclui o processo 25 minutos mais cedo. Quando o uso do ciclo das prensas é comparado, que são as principais causas de *bottlenecks*, esta solução usa 5 ciclos, ao contrário das 6 usadas na abordagem FIFO. A média da carga das prensas quando é iniciado o

processamento é de 65,4%, contra 54,67%.

Considerando o tempo de espera dos condutores, nesta solução os condutores esperam um total de 245 minutos (108 minutos a menos que a abordagem FIFO), ou 40.83 minutos em média (contra 58.83). Isto é, com esta solução os condutores esperam em média menos 18 minutos para começar a descarregar as uvas. Quando estes tempos de espera por variedade de uva são analisados, é possível notar que os condutores que carregam as uvas Fernão Pires, as mais frágeis, esperam em média 1 minuto (sendo que ambos começam a descarga no inicio do processo) contra 78.5 minutos. Os condutores que trazem uvas Loureiro e Misc esperaram em média 84.7 e 44 minutos, respetivamente (contra 52 e 27 minutos na abordagem FIFO). Isto mostra, como esperado, um aumento do tempo de espera dos que têm tipos de uvas menos frágeis, apesar da média dos tempos de espera globais terem descido significativamente. As Figuras 25 e 26 mostram como estes tempos mudaram ao longo da evolução do GA.

O fitness desta solução, quando avaliada com a mesma função de fitness, é de 33.2. A Figura 27 mostra a evolução da média de fitness de cada geração, assim como o fitness das melhores soluções encontradas a qualquer momento. A Figura mostra que o GA convergiu depois de 1000 iterações.

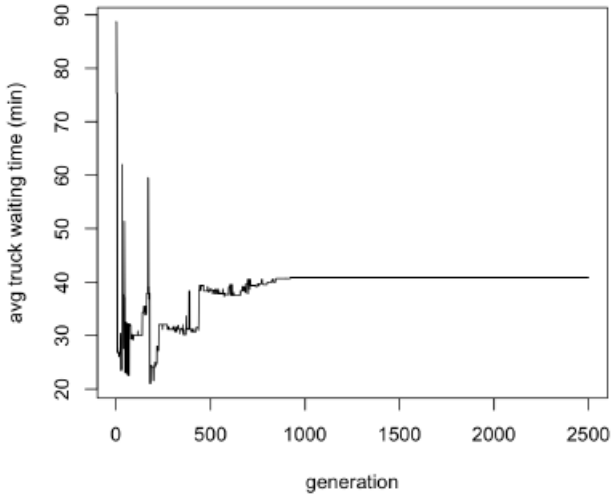


Figura 25: Variação da média do tempo de espera de cada condutor ao longo da evolução do GA

A diferença mais impressionante quando comparando as soluções é a priorização das uvas mais frágeis independentemente da ordem de chegada dos camiões. É interessante também que na solução do GA, existem três momentos em que ambos os tegões estão a enviar uvas para a mesma prensa. Portanto, as soluções têm a tendência de tratar camiões carregados com o mesmo tipo de

uvas simultaneamente, e enviar as suas uvas para uma única prensa de modo a enchê-la o mais rápido possível, de modo a dispensar os camiões. Outro aspeto digno de destaque é o começo do processamento da P_2 muito cedo, a uma capacidade de apenas 35%, e enquanto recebia uvas de um tegão. Isto é possível pois as restantes uvas que estão no sistema (em ambos os tegões) todas elas cabem na outra prensa, permitindo que o primeiro comece a processar o mais cedo possível.

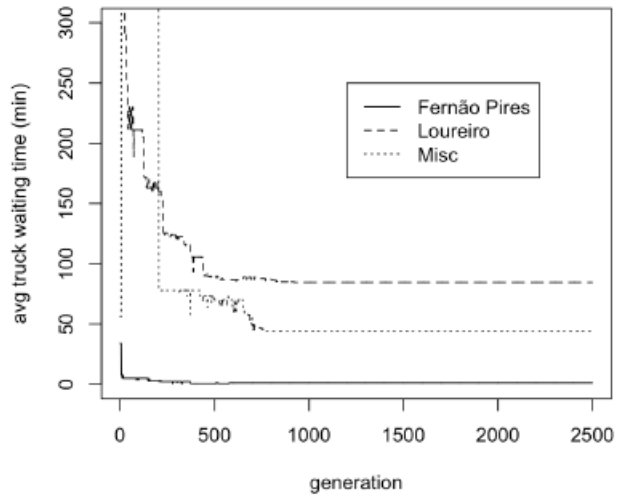


Figura 26: Variação da média do tempo de espera de cada condutor tendo em conta a variedade da uva ao longo da evolução do GA

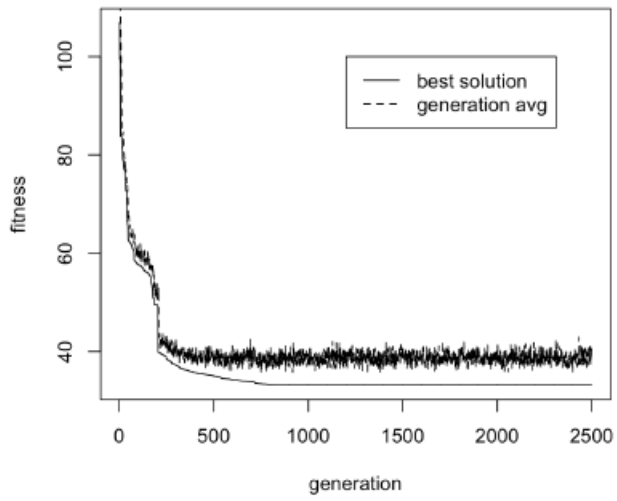


Figura 27: Evolução do fitness durante a execução do GA

4.3 Avaliação

Os GA dependem significativamente do acaso, tanto durante a inicialização, como durante a reprodução. Portanto, uma única execução do algoritmo não representa a sua performance. Por esta razão, nesta Secção o algoritmo foi executado 10 vezes e foram medidos alguns indicadores chave, como forma de avaliar a adequação do algoritmo proposto para gerar soluções válidas num tempo útil. No entanto, e dado o tamanho das soluções deste tipo de problemas (como demonstrado nas tabelas 2 e 3), foi decidido apenas detalhar estes indicadores de performance.

Estas 10 execuções do algoritmo foram executadas num sistema com um processador 2.7GHz Intel Core i7 e uma RAM de 16 GB 2133 MHz. O mesmo cenário, como descrito anteriormente, foi usado em 10 execuções. O algoritmo foi configurado com uma população de 50 soluções em cada geração, e a reprodução foi configurada como detalhado na Secção 4.2. Como critérios de paragem, considera-se um máximo de 2500 gerações ou quando ocorre convergência. Considerando que a convergência é atingida quando o fitness da melhor solução encontrada não muda ao longo de 20 gerações consecutivas.

No que diz respeito ao fitness das 10 melhores soluções encontradas, a média é de 34 ± 2.54, como pode ser visualizado na Tabela 4. O fitness da primeira execução do GA detalhado na Secção 4.2 foi de 33.2, que está dentro deste intervalo. Em comparação, o fitness da abordagem FIFO foi de 58.83, que é significativamente maior, como detalhado na Secção 4.1. O valor relativamente baixo de σ também indica que este é um processo relativamente viável pois a variação do fitness das soluções encontradas é baixo. A Figura 28 combina a evolução do fitness ao longo do tempo para as 10 execuções.

	$\overline{x}$	$\tilde{x}$	σ	min	max
Fitness	34.22	33.48	2.54	31.93	40.91
Convergência (gerações)	1407.6	1315.5	422.00	648	2224
Tempo de execução (s)	146.5	140.5	46.20	62	235
Média do tempo de espera de um camião (min.)	44.25	21.58	11.26	30.17	61.17

Tabela 4: Estatísticas dos quatro indicadores chave ao longo das 10 execuções do GA.

Quando considerando a ocorrência de convergência, em média após 1400 ± 422 iterações, que correspondem, em média, a pouco mais de 2 minutos. Embora isto não seja excepcionalmente baixo, permite que o algoritmo seja usado em tempo real, sempre que um novo camião chega e enquanto este está a ser pesado e o alcool das suas uvas está a ser medido. Na execução mais curta, a convergência foi atingida depois de 648 iterações (62 segundos) enquanto que na execução mais longa convergiu depois de 2224 iterações (perto de 4 minutos).

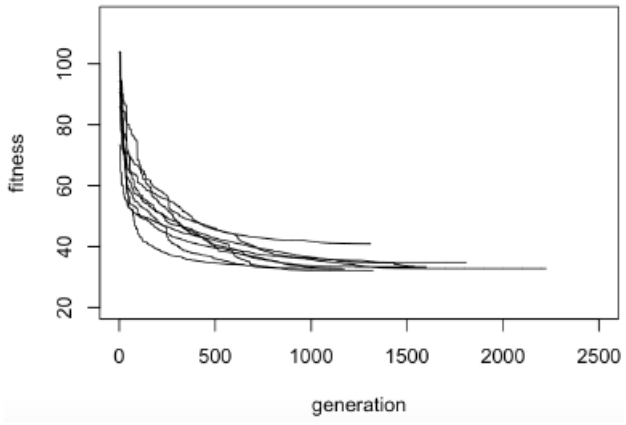


Figura 28: Evolução do fitness durante as 10 execuções do GA

Finalmente, quando comparando o tempo de espera do camião, em média cada condutor iria esperar à volta de 44 $\pm$ 11 minutos. Isto é bem menos do que os 60 minutos que os condutores teriam de esperar para o mesmo cenário, em média, quando usada a abordagem FIFO, como detalhado na Secção 4.1.

4.4 Aplicação do GA a um cenário real

Nesta Secção é detalhada uma solução proposta pelo GA para um cenário real do problema, sendo que deste modo é possivel mostrar o comportamento do algoritmo num contexto real. Para isso era necessário possuir dados reais sobre uma adega de uma empresa e portanto foi escolhida a empresa Sogrape. Foi feito um pedido para saber se seria possível o fornecimento de dados relativos à mesma, sendo que estes foram prontamente enviados, o que contribuiu para que esta simulação fosse realizada com ainda mais rigor e com um cenário real.

Como efectuado na Secção 4.3 em que foi analisada a solução GA genérica, foram efetuadas 10 simulações de modo a que fosse possivel medir a performance e indicadores chave, sendo estes os mesmos. As 10 execuções foram efetuadas num sistema com processador 2.8Ghz Intel Core I7 e uma RAM de 8GB.

O GA foi configurado da seguinte maneira. Uma população de 50 indivíduos foi usada, com uma duração de 480 minutos cada (8 horas). Em cada iteração as melhores 25 são selecionadas para serem reproduzidas, sendo que cada uma das soluções selecionadas tem a mesma probabilidade de mutar ou serem cruzadas entre si aleatoriamente. Cada nova geração é gerada através das melhores 25 da geração anterior mais a 25 novas soluções geradas através dos operadores de reprodução. O

algoritmo pára assim que haja convergência ou depois de atingir 2500 iterações.

A Sogrape, mais especificamente o Centro de Vinificação de Anadia da Sogrape, durante as vindimas deste ano, 2019, registou 1998 camiões a entrarem na adega durante o período de colheitas (época alta) que durou de 12 de setembro a 7 de outubro, ou seja 17 dias. Em média foram 117 camiões por dia, cada um com uma carga média de 2,341 toneladas, sendo que durante este período foram recebidas e processadas 4678 toneladas de uvas. O centro possui 4 tegões, cada um com a capacidade de 10 m^3, ou seja 10 toneladas cada um, sendo estes capazes de processar 30 toneladas de uvas por hora. Quanto às prensas, existem 4 cada uma capaz de processar 45 toneladas de uvas por hora. Este centro de vinificação é um dos maiores da Sogrape, sendo um ótimo exemplo para testar o comportamento do GA num cenrário real.

Camião	Uva	Fragilidade	Carga	Tipo
T_1	Touriga Nacional	0.8	2341 Kgs	basculante (2 ton./min)
T_2	Rufete	0.4	2341 Kgs	basculante (2 ton./min)
T_3	Tinta Barroca	0.5	2341 Kgs	basculante (2 ton./min)
T_4	Touriga Nacional	0.8	2341 Kgs	basculante (2 ton./min)
T_5	Rufete	0.4	2341 Kgs	não basculante (0.5 ton./min)
T_6	Baga	0.2	2341 Kgs	não basculante (0.5 ton./min)
T_7	Tinta Barroca	0.5	2341 Kgs	não basculante (0.5 ton./min)

Tabela 5: Características dos camiões do cenário real.

Como referido anteriormente, por dia em média entram 117 camiões no centro e para este cenário foi simulada uma das horas do dia. Durante o período de vindimas, o centro costuma trabalhar durante 16 horas por dia, de modo a minimizar o intervalo de tempo entre a colheita e o processamento das uvas. tendo isto em conta para esta simulação foi usada a quantidade média de camiões por hora (117 camiões a dividir por 16 horas são aproximadamente 7 camiões por hora). Nesta simulação foram selecionadas quatro casta de uvas diferentes, uvas para produzir diferentes vinhos, entre eles marcas como Mateus Rosé e Porto Ferreira estas castas são: Baga, Rufete, Tinta Barroca e Touriga Nacional. Na tabela 5 são mostradas as características dos 7 camiões da simulação.

Nas figuras 29, 30 e 31 são mostradas as cargas das três entidades ao longo do dia. Através destas o utilizador tem uma percepção clara do que acontece a cada entidade ao longo do tempo.

Na Figura 29 é possível verificar que os camiões realizam o processo de checkout entre o minuto 5 e 30 após a sua chegada, havendo uma média de aproximadamente 14 minutos até que estes realizem o checkout.

Uma das características que salta logo à vista na execução deste cenário, foi a escolha de apenas

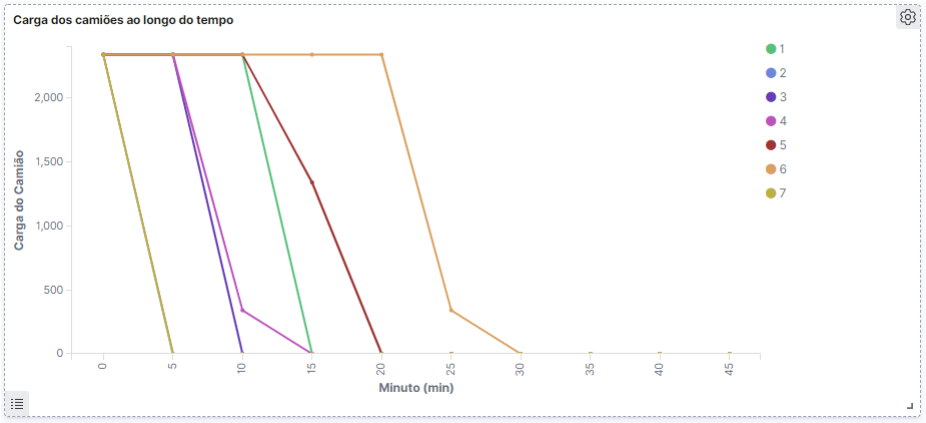


Figura 29: Carga dos camiões ao longo da última simulação do cenário real

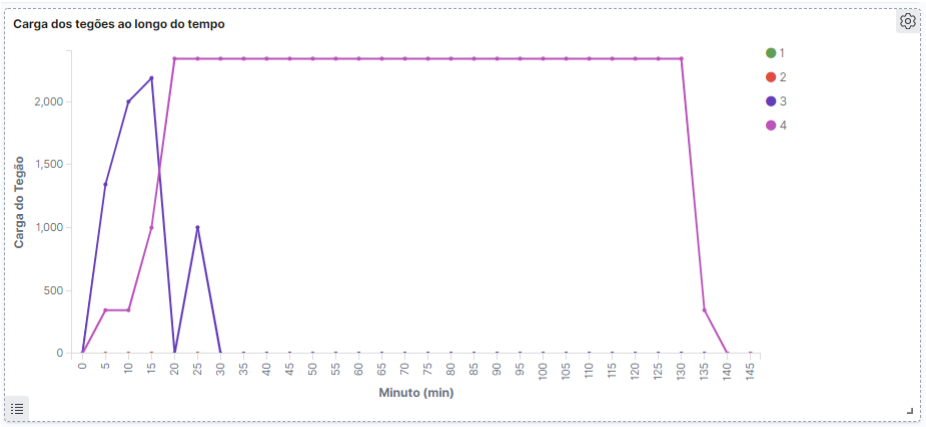


Figura 30: Carga dos tegões ao longo da última simulação do cenário real

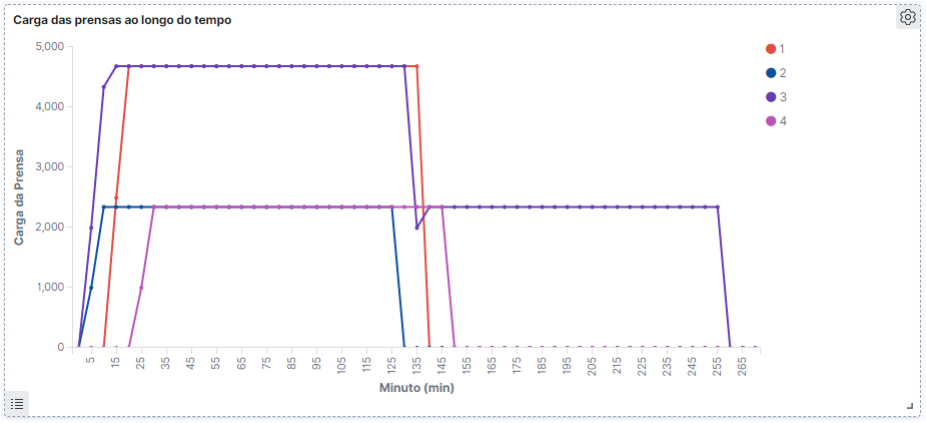
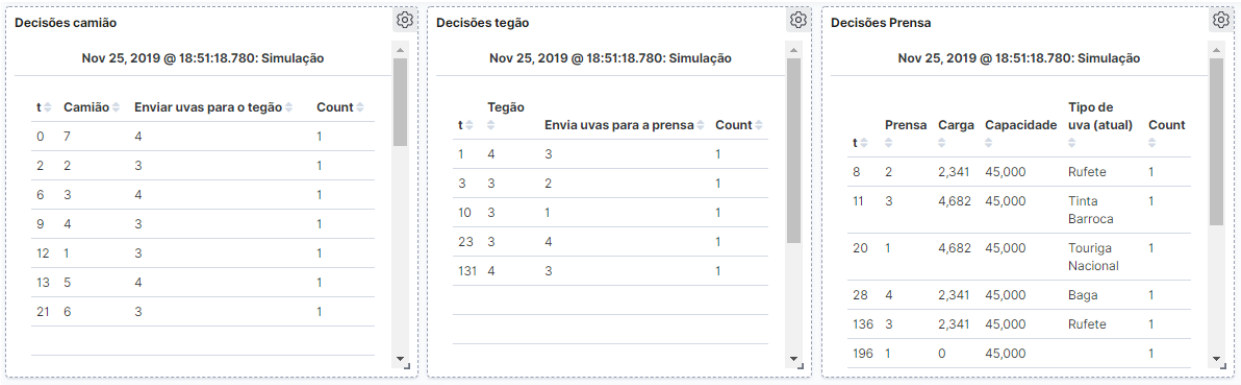


Figura 31: Carga dos prensas ao longo da última simulação do cenário real

dois tegões para a resolução do problema neste cenário. Como pode ser visualizado na Figura 32 foi escolhido preferencialmente os tegões três e quarto para o envio das uvas para as respetivas prensas.



Decisões camião

Nov 25, 2019 @ 18:51:18.780: Simulação

t	Camião	Enviar uvas para o tegão	Count
0	7	4	1
2	2	3	1
6	3	4	1
9	4	3	1
12	1	3	1
13	5	4	1
21	6	3	1

Decisões tegão

Nov 25, 2019 @ 18:51:18.780: Simulação

t	Tegão	Envia uvas para a prensa	Count
1	4	3	1
3	3	2	1
10	3	1	1
23	3	4	1
131	4	3	1

Decisões Prensa

Nov 25, 2019 @ 18:51:18.780: Simulação

t	Prensa	Carga	Capacidade	Tipo de uva (atual)	Count
8	2	2,341	45,000	Rufete	1
11	3	4,682	45,000	Tinta Barroca	1
20	1	4,682	45,000	Touriga Nacional	1
28	4	2,341	45,000	Baga	1
136	3	2,341	45,000	Rufete	1
196	1	0	45,000		1

Figura 32: Decisões tomadas pela solução mais apta encontrada pelo Algoritmo Genético para o cenário real

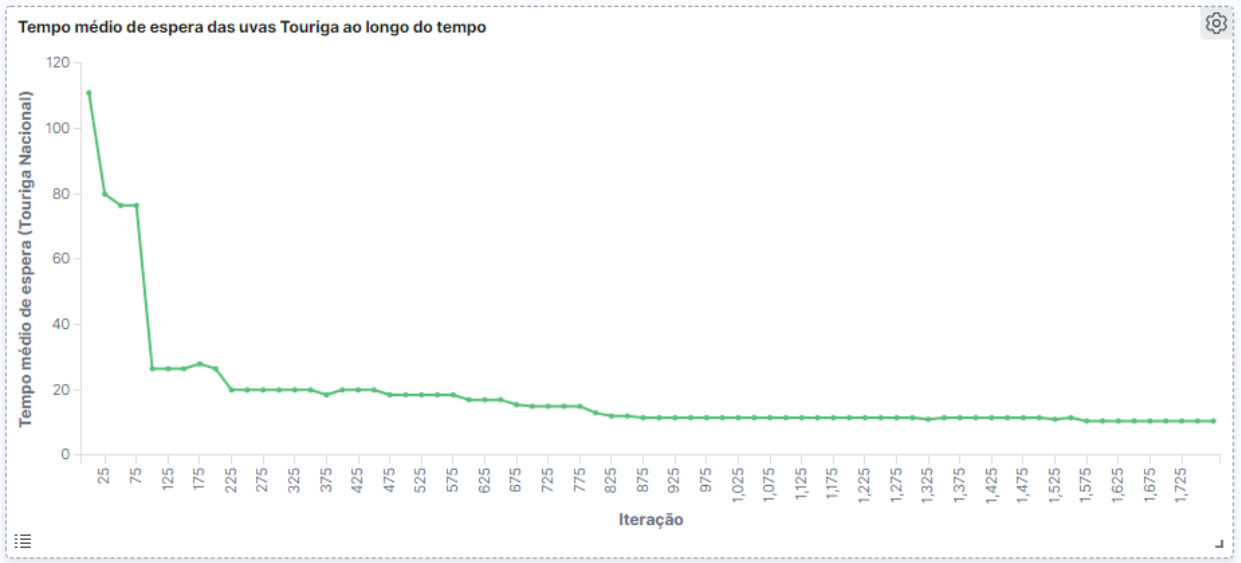


Figura 33: Tempo médio de espera das uvas Touriga Nacional ao longo das iterações do GA (cenário real)

Como é possível visualizar através da análise das figuras 35, 34, 36 e 33, o tempo médio de espera dos camiões de cada tipo de uvas vai diminuindo ao longo das iterações do GA, à medida que este vai selecionando soluções cada vez mais aptas. Se forem analisados os valores das prioridades de cada tipo de uva na tabela 5 é possível verificar que as uvas com maior prioridade são as

Touriga Nacional seguidas pelas Tinta Barroca. Ao analisar os quatro gráficos é possível verificar que estas são as que possuem menor tempo de espera ao longo das iterações do Algoritmo, fazendo com que o impacto do tempo de espera das mesmas seja minimizado.

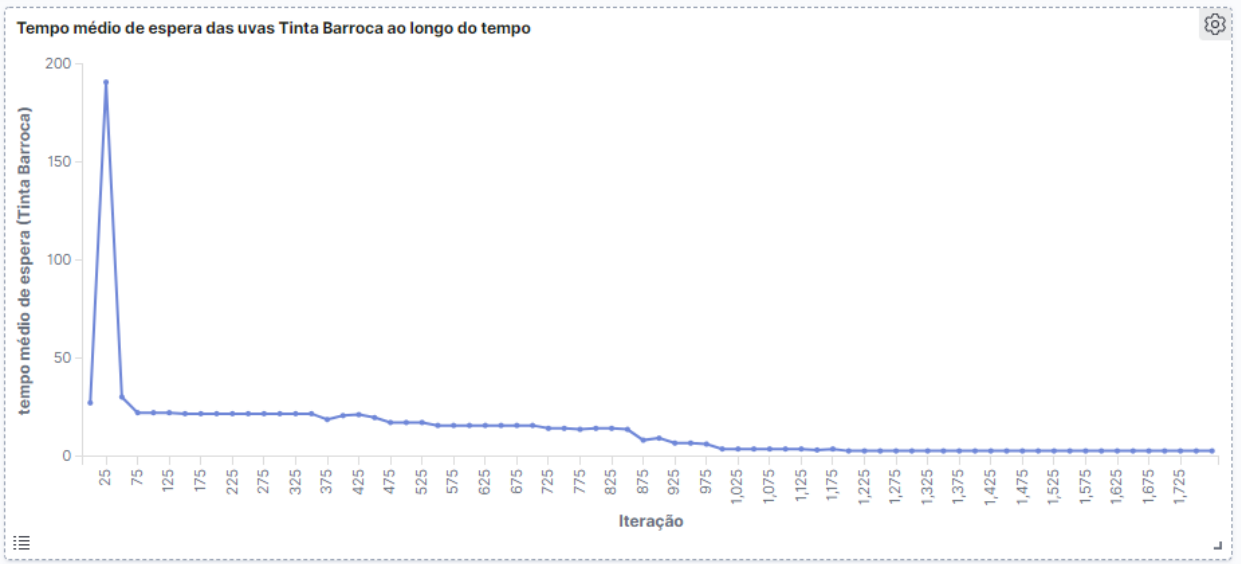


Figura 34: Tempo médio de espera das uvas Tinta Barroca ao longo das iterações do GA (cenário real)

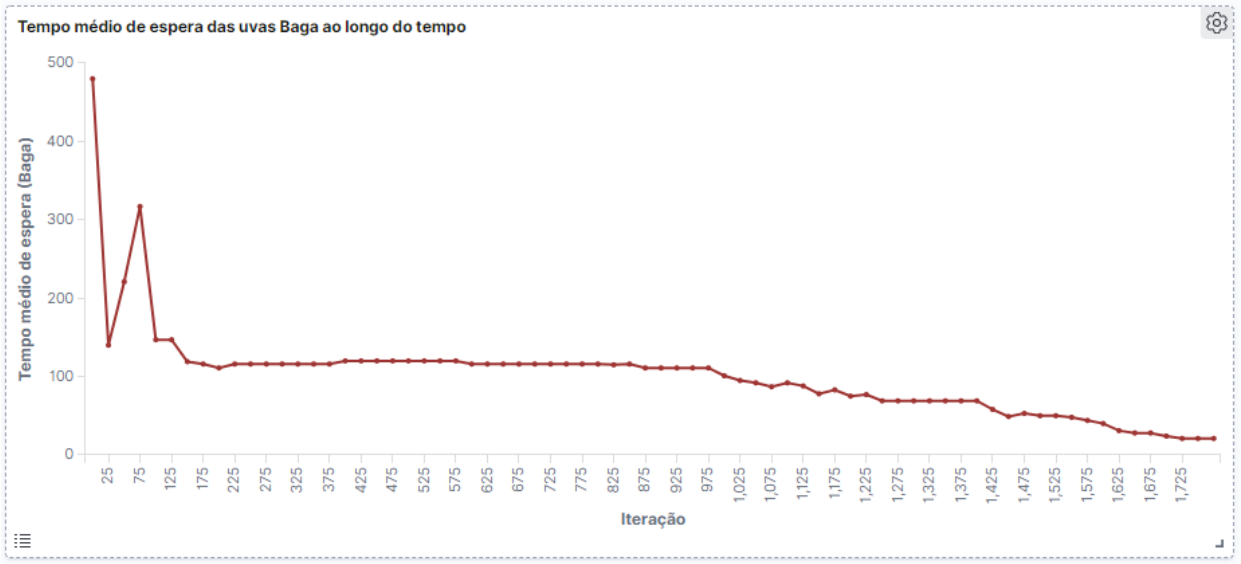


Figura 35: Tempo médio de espera das uvas Baga ao longo das iterações do GA (cenário real)

Por outro lado, nos gráficos referentes às médias de tempo de espera das uvas Baga e Rufete, que podem ser visualizados nas figuras 35 e 36 respetivamente, é possível verificar que os

seus tempos de espera também vão diminuindo ao longo do tempo, mas ao aproximarem-se das iterações finais, estes não possuem valores tão baixos quanto os dois tipos anteriormente referidos.

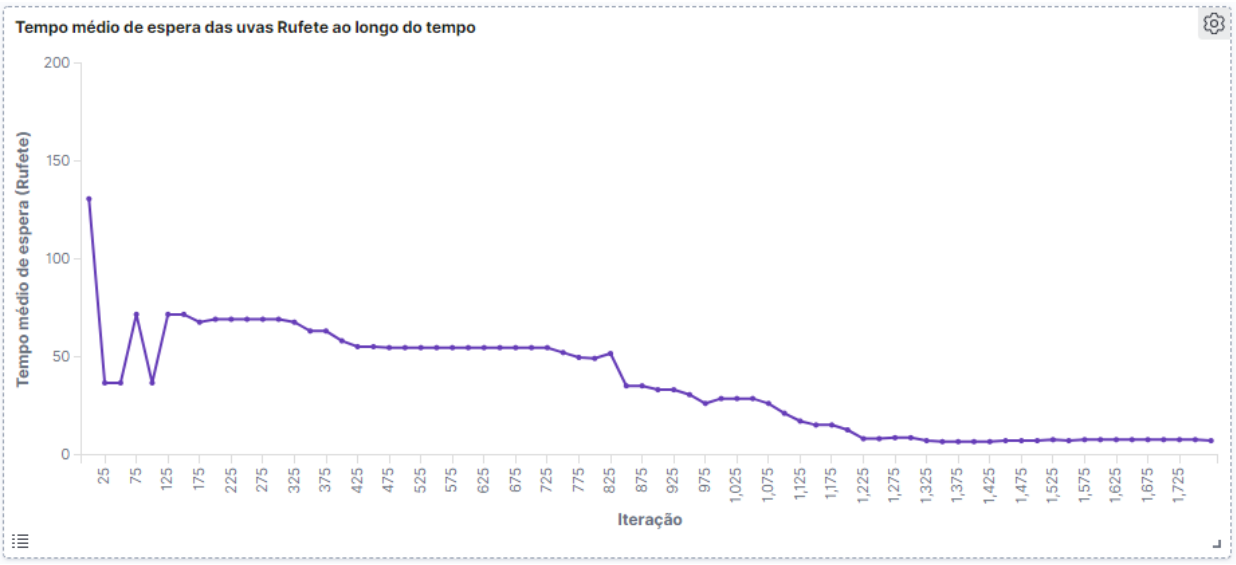


Figura 36: Tempo médio de espera das uvas Rufete ao longo das iterações do GA (cenário real)

Por fim como pode ser verificado no gráfico da média do fitness do GA, que pode ser visualizado na Figura 37, é possível ver que ao longo das iterações do GA o fitness vai diminuindo ao longo do tempo, no entanto ao chegar perto das 1000 iterações este mantém-se relativamente constante. Apesar deste parecer estagnado ao longo de tantas iterações, este ainda não convergiu, ou seja o seu fitness ainda continua a ser alterado.

As 10 execuções do GA foram executadas e em cada uma delas foram extraídos um conjunto de dados de modo a ser possível analisar a performance do mesmo ao longo da sua execução. Através desses dados foi possível a construção da tabela 6 em que podem ser analisadas algumas métricas acerca das dez execuções.

Relativamente ao fitness das 10 melhores soluções encontradas, a média é de 7.352 ± 1.974. Infelizmente não existem dados relativos ao tempo de espera de cada camião, mas sabe-se que atualmente, em média, cada camião demora aproximadamente 50 minutos entre o processo de checkin e checkout. Na Tabela 29 é possível verificar que na simulação deste cenário demoraram entre 5 a 30 minutos a realizar o processo de checkout, sendo a média de aproximadamente 14 minutos. Se forem descartadas as variáveis externas que poderão ocorrer no dia-a-dia do centro e que atrasem o processo, é possível dizer que de 50 minutos o GA conseguiu otimizar o processo para 13, ou seja 37 minutos mais rápido. Estes novos valores, trazem uma vantagem muito grande se existirem dentro da empresa, e farão com que muitos mais camiões possam ser processados

dentro dos 50 minutos médios atualmente existentes.

	$\overline{x}$	$\tilde{x}$	σ	min	max
Fitness	7.352	6.715	1.974	5.418	12.548
Convergência (gerações)	1493.667	1425	188.272	1225	1841
Tempo de execução (s)	169.283	157.946	24.129	142.007	219.256

Tabela 6: Estatísticas dos quatro indicadores chave ao longo das 10 execuções do Algoritmo Genético no cenário real.

De seguida, ao analisar o tempo de execução do algoritmo, é possivel verificar que a sua média é de 169.283 seg. $\pm$ 24.129 seg., ou seja, em média a execução do algoritmo demora 2.82 minutos. Se for subtraído aos 37 minutos que a empresa ganha ao usar o GA os 2.82 minutos que este demora a executar, a empresa consegue fazer com que os seus camiões realizem o processo de checkout 34 minutos mais rápido que atualmente. Estes valores foram a atingidos com os recursos computacionais referidos anteriormente, mas com um maior investimento por parte das empresas, estes valores tornar-se-ão menores.

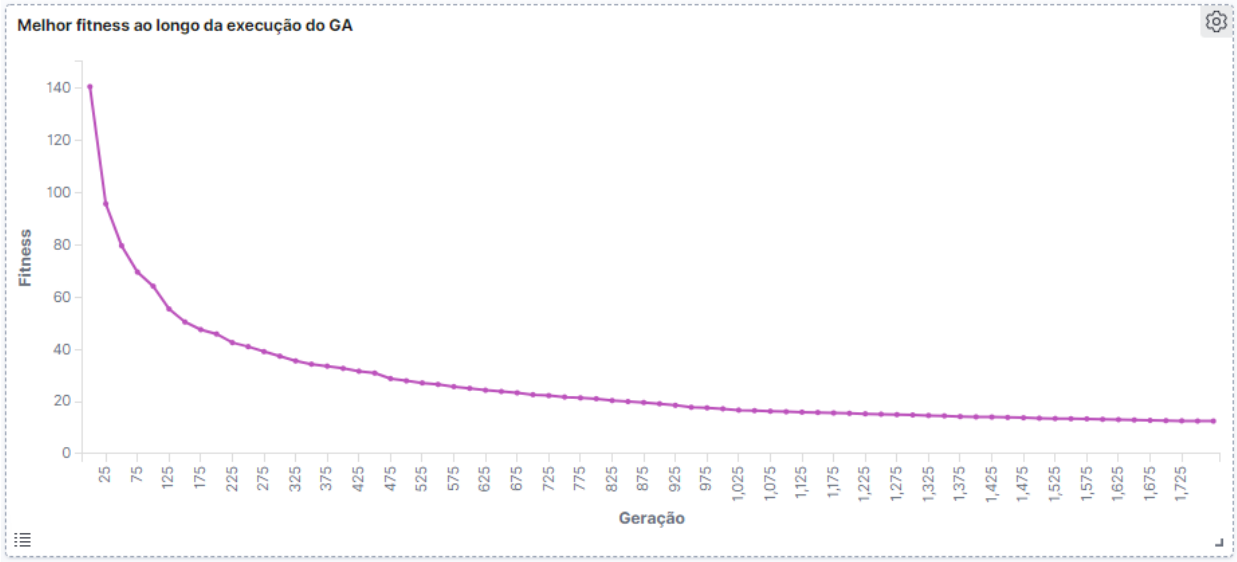


Figura 37: Fitness das melhores soluções encontradas pelo GA ao longo da sua execução (cenário real)

5 Conclusões

Portugal é um país com uma longa tradição vinícola, tendo iniciado a sua história com a implantação da vinicultura pelos romanos. Desde então, tem vindo a evoluir e a produzir vinhos de cada vez melhor qualidade. Nas última décadas a vinicultura portuguesa tem evoluído cada vez mais tecnologicamente, como consequência do importante desenvolvimento económico, social e politico do país. Com a ajuda da tecnologia, castas que originavam vinhos de qualidade inferior, passaram a dar grandes vinhos, aperfeiçoando as suas características únicas.

O problema das adegas descrito nesta dissertação, afeta diretamente a qualidade das suas uvas, visto que quanto mais tempos estas esperam para ser processadas, mais elas se deterioram. Como demonstrado na Secção 4.4 as empresas fazem um esforço notório durante as épocas de colheita, de modo a que haja uma minimização do tempo de espera das uvas entre a sua colheita e o seu processamento. No cenário real descrito, os colaboradores da Sogrape trabalham 16 horas por dia durante os 17 dias de colheitas para reduzir ao máximo este intervalo de tempo. No entanto na segunda fase do processo, receção e processamento das uvas na adega, os colaboradores, usam a abordagem FIFO e intuição adquirida ao longo dos vários anos de experiência para gerir este processo.

De modo a resolver este problema vários algoritmos de otimização inspirados na natureza (NABI) foram estudados. Devido ao número de aplicações encontradas no levantamento bibliográfico sobre algoritmos GA, DE, SA, HS, PSO, ACO, FA e BA, foi feita a revisão mais detalhada de cada um deles. Existem muitos outros, estando estes apenas mencionados na Secção 2.1.9. Após o estudo e análise dos algoritmo anteriormente descritos, para este projeto os Algoritmos Genéticos foram escolhidos devido à sua fácil implementação, necessidade de poucas configurações, mas também pelo facto de não existirem documentados, projetos relacionados com este problema utilizando este algoritmo.

Dois casos de estudo fora realizados, um deles utilizando a abordagem FIFO, simulando aproximadamente a abordagem utilizada atualmente pelas adegas na receção e processamento das uvas dos camiões dos seus produtores e outra utilizando o GA proposto.

No primeiro caso de estudo a lógica FIFO é usada, esta simula aproximadamente o comportamento atual das adegas, processando os camiões pela ordem de chegada, no entanto podem existir alguns casos em que a intuição do profissional que gere a receção dos camiões pode mudar um pouco esta lógica. No que diz respeito à avaliação da possível solução FIFO, os condutores esperaram um total de 353 minutos, ou seja uma média de 58,83 minutos por condutor. As uvas mais frágeis, Fernão Pires, esperaram uma média de 78,5 minutos, enquanto que condutores que

carregavam uvas do tipo Loureiro e Misc esperaram, em média, 52 e 27 minutos respetivamente.

No segundo caso de estudo foi detalhada uma solução descoberta pelo Algoritmo Genético proposto para o mesmo cenário estudado na abordagem FIFO. A configuração do GA é detalhada na Secção 4.2. em comparação com a abordagem FIFO, a solução gerada automáticamente possui uma decisão a mais (19 decisões). No entanto conclui o processo 25 minutos mais cedo. Quando o uso do ciclo das prensas é comparado, que são as principais causas de *bottlenecks*, esta solução usa 5 ciclos, ao contrário das 6 usadas na abordagem FIFO. A média da carga das prensas quando é iniciado o processamento é de 65,4%, contra 54,67%.

Considerando o tempo de espera dos condutores, nesta solução os condutores esperam um total de 245 minutos (108 minutos a menos que a abordagem FIFO), ou 40,83 minutos em média (contra 58,83). Isto é, com esta solução os condutores esperam em média 18 minutos a menos para começar a descarregar as uvas. Quando estes tempos de espera por variedade de uva são analisados, é possível notar que os condutores que carregam as uvas Fernão Pires mais frágeis esperam em média 1 minuto (sendo que ambos começam a descarga no início do processo) contra 78.5 minutos. Os condutores que trazem uvas Loureiro e Misc esperaram em média 84.7 e 44 minutos, respetivamente (contra 52 e 27 minutos na abordagem FIFO). Isto mostra, como esperado, um aumento do tempo de espera dos que têm tipos de uvas menos frágeis, apesar da média dos tempos de espera terem descido significativamente.

A diferença mais impressionante quando comparando ambas as soluções é a priorização das uvas mais frágeis independentemente da ordem de chegada dos camiões. É interessante também que na solução do GA, existem 3 momentos em que ambos os tegões estão a enviar uvas para a mesma prensa. Portanto, as soluções têm a tendência de tratar camiões carregados com o mesmo tipo de uvas simultaneamente, e enviar as suas uvas para uma única prensa de modo a enchê-la o mais rápido possível, de modo a dispensar os camiões. Outro aspeto digno de destaque é o começo do processamento da P_2 muito cedo, a uma capacidade de apenas 35%, e enquanto recebia uvas de um tegão. Isto é possível pois as restantes uvas que estão no sistema (em ambos os tegões) todas elas cabem na outra prensa, permitindo que o primeiro comece a processar o mais cedo possível.

Por fim, e de modo a mostrar o comportamento do algoritmo num contexto real, foi pedido à empresa Sogrape o fornecimentos de dados relativo a uma das suas adegas. Esta prontamente forneceu dados relativos às vindimas deste ano (2019) e dados relativos ao Centro de Vinificação de Anadia da Sogrape. Com estes foi possivel realizar a simulação de um cenário real. A empresa referiu que em média um camião demora cerca de 50 minutos entre o processo de checkin e checkout. Se compararmos estes valores com os obtidos pela solução descoberta pelo GA e descartando variáveis externas que poderão ocorrer no dia-a-dia do centro, atrasando o processo,

conseguimos verificar que o GA conseguiu otimizar o processo para 13 minutos em média, ou seja 37 minutos mais rápido. Se retirarmos os 2,82 minutos médios de execução, obtidos durante as 10 execuções do cenário real estando referidos na Secção 4.4, podemos dizer que o algoritmo otimizou o processo em 34 minutos. Estes valores trazem uma vantagem muito grande se existirem dentro da empresa e farão com que muitos mais camiões possam ser processados dentro dos 50 minutos médios atualmente existentes.

Tendo em conta as análises efetuadas anteriormente, é possível dizer que com o uso deste Algoritmo Genético, as produtoras de vinho são capazes de melhorar os seu processo de escalonamento dos camiões. Este trará vantagens tanto ao nível de logística, com a otimização dos espaços usados e tempos de espera, como também indirectamente em termos económicos, com a garantia do mantimento da qualidade das uvas antes do seu processamento, isto comparativamente com a abordagem FIFO utilizada atualmente pelas adegas. Através do cenário simulado com a ajuda de dados reais do Centro de Vinificação de Anadia da Sogrape, o GA foi capaz de otimizar em 34 minutos o tempo de espera dos camiões dos produtores de vinho da empresa.

Através dos *dashboards* criados no Kibana, o utilizador é capaz de aceder a informações detalhadas sobre a solução mais apta encontrada pelo GA. Esta pode ser gerada com um tempo de execução entre 2 a 3 minutos por simulação. Assim, com os *dashboards* o utilizador consegue saber exatamente como distribuir os camiões pelas diferentes entidades, assim como obter informações sobre como se deve comportar cada uma delas, de modo que se garanta assim a maximização da otimização do processo.

5.1 Principais Resultados

Como citado anteriormente, um dos objetivos com a elaboração desta dissertação é ajudar as adegas durante o processo de escalonamento e processamento das suas uvas e neste sentido ajuda-las a otimizar recursos, lucros e a garantir a qualidade dos seus vinhos. Neste sentido, e devido à escassez de artigos relacionados com este tema, envolvidos com algoritmos de otimização, houve um esforço na apresentação de um poster e na escrita e submissão de dois artigos científicos tendo um deles sido submetido na revista *Neural Computing and Applications*, que possui um fator de impacto de 4.664. Os dois artigos submetidos, um deles já aceite para publicação, e o poster são detalhados de seguida:

- **Carneiro D.**, Pereira J., Silva E.C. (2019) Optimization of the grapes reception process. Submitted to Neural Computing and Applications (Submetido).

  URL `https://www.springer.com/journal/521`

Abstract Grapes reception is a key process in wine production. The harvest days are extremely challenging days in managing the reception of the grapes. In fact, the winery needs to deal with the non-uniform arrival of the grapes, while guaranteeing suppliers' satisfaction and wine quality. The best management of the resources of the suppliers (i.e. grapes and trucks) and winery (i.e. grain-tanks and pressing machines) must be ensured. In this paper the underlying optimization problem for grape reception is solved by developing a Genetic Algorithm tailored for this specific challenge. The results of this algorithm are compared with a FIFO policy for a typical scenario that occurs on the harvest days of a real winery. The results show that, using modest computational resources, the convergence of the algorithm is, on average, two minutes. This allows for the algorithm to be used in real-time, since this optimization problem may be solved whenever a new truck arrives and while it is being weighted and the grape's alcohol content measured. Furthermore, the trucks waiting time for the results using the developed GA are significantly smaller than the one sobserved using a FIFO approach.

- **Pereira J.**, Carneiro D., Silva E.C. (2019) Otimização do Processo de Receção e Processamento de Uvas em Lagares. In: SEI - Simpósio de Engenharia Informática

  Abstract Atualmente e principalmente durante os períodos de colheitas, as empresas produtoras de vinho, deparam-se com os seus produtores a enfrentar longos períodos de espera para realizar a descarga das suas uvas, o que provoca um impacto negativo na qualidade dos vinhos. O objetivo deste projeto é de desenvolver um Algoritmo Genético capaz de, através de um conjunto de *inputs* iniciais, encontrar a melhor solução para a otimização do escalonamento dos camiões dos produtores dessas empresas. Deste modo haverá um melhor uso da capacidade de produção disponível, otimização do espaço utilizado durante todo o processo e diminuição do tempo de espera dos produtores, evitando assim, a perda de qualidade das suas uvas. Os resultados do algoritmo poderão ser consultados pelo utilizador através de visualizações gráficas dos mesmo.

- **Silva E.C.**, Pereira J., Carneiro D. (2019, Junho). Optimization of the Grapes Reception Process. Poster session presented at the VI Workshop on Computational Data Analysis and Numerical Methods (WCDANM), Departamento de Matemática da Universidade da Beira Interior (UBI), Covilhã.

5.2 Trabalho futuro

Os próximos passos no desenvolvimento deste projeto passarão por simular e aplicar os resultados do GA numa adega real, deste modo será possível ter um *feedback* mais profundo através da utilização do mesmo por parte dos utilizadores, assim poderá será possível detetar possíveis pontos de melhoria.

Para que o primeiro ponto seja realizado com mais facilidade por parte dos utilizadores, um outro ponto passa pela criação de uma aplicação onde poderá ser possível configurar todas as variáveis do GA e também simular o cenário na mesma. Várias opções podem ser tidas em conta neste ponto, visto que esta solução de aplicação pode ser adaptada à realidade das empresas. Como descrito na Secção 3.1 as empresas normalmente possuem um sistema de ERP onde amazenam todas as informações relativas à mesma, estando entre elas também as variáveis necessárias para a configuração do GA. No entanto, de empresa para empresa estes ERP podem ser diferentes, e para evitar que o utilizador seja obrigado a configurar o GA a cada simulação, uma estratégia possivel, seria a criação de um *middleware* que seria responsável por integrar cada um dos ERP e configurar automaticamente o GA e realizar uma nova simulação sempre que uma mudança nas variáveis ocorresse. Isto iria acelerar o processo pré-simulação, o que iria minimizar o tempo necessário para a obtenção de uma solução, aproximando a obtenção da mesma do tempo real.

Por fim, um último ponto passaria pelo desenvolvimento de uma aplicação móvel que permitiria aos colaboradores visualizarem as soluções enquanto se movimentam pelo parque. Estes também poderiam interagir com a mesma, sendo que o algoritmo adaptar-se-ia caso uma decisão diferente da lista apresentada fosse tomada, não existindo a necessidade do colaborador configurar novamente o algoritmo para o seu novo cenário.

6 Bibliografia

[1] (2019, 11). [Online]. Available: https://www.clubevinhosportugueses.pt/vinhos/como-se-produz-vinho-rececao-de-uva-desengace-esmagamento-e-esgotamento-4/

[2] (2019, 11). [Online]. Available: https://tienda.todobodega.com/prensa-neumatica-della-toffola-pe-100-para-100-hl/

[3] (2019, 11). [Online]. Available: https://www.mundoportugues.pt

[4] (2019, 11). [Online]. Available: https://esa.github.io/pagmo2/docs/cpp/algorithms/pso.html

[5] (2019, 11). [Online]. Available: https://plos.figshare.com/articles/_Ant_Colony_Optimization_Algorithm_processes_/1418788/1

[6] (2019, 11). [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-67669-2_9

[7] (2019, 11). [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/62235-firefly-feature-selection-and-optimization?focused=7445925&tab=function

[8] D. Contexto, Cenários, M. Leonor, M.-L. Assad, and J. Almeida, "Agricultura e sustentabilidade," 11 2019.

[9] I. N. de Estatística, "Estatísticas agrícolas : 2018. lisboa : Ine," 2019.

[10] (2019, 11). [Online]. Available: https://www.academiadovinho.com.br/__mod_regiao.php?reg_num=PT

[11] (2019, 11). [Online]. Available: https://www.linkedin.com/pulse/intelig%C3%AAncia-artificial-e-iot-na-agricultura-adriano-naspolini/

[12] (2019, 11). [Online]. Available: https://www.aveleda.com/

[13] (2019, 11). [Online]. Available: https://www.sograpevinhos.com/

[14] T. Back, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*, 1st ed. Bristol, UK, UK: IOP Publishing Ltd., 1997.

[15] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms: Second Edition*. Luniver Press, 2010.

[16] R. A. Blum, C., "Metaheuristics in combinatorial optimization: overview and conceptual comparison," pp. 268–308, 2013.

[17] X. Yang, "Engineering optimization: An introduction with metaheuristic applications," 2010.

[18] P. G. Nanda, S.J., "A survey on nature inspired metaheuristic algorithms for partitional clustering," pp. 1–18, 2014.

[19] S. M. C. S. S. Y. Hussain, K., "Metaheuristic research: a comprehensive survey." pp. 1–43, 2018.

[20] J. Holland, "Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence," 1992.

[21] P. K. Storn, R., "Differentialevolution-asimpleandefficientheuristicforglobal optimization over continuous spaces," 1995.

[22] G. C. V. M. Kirkpatrick, S., "Optimization by simulated annealing," pp. 671–680, 1983.

[23] K. J. L. G. Geem, Z.W., "A new heuristic optimization algorithm: harmony search," pp. 39–43, 2001.

[24] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Oct 1995, pp. 39–43.

[25] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Milano, Italy, 1992.

[26] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms: Second Edition*. Luniver Press, 2010.

[27] X. Yang. (2010) A new metaheuristic bat-inspired algorithm. [Online]. Available: https://doi.org/10.1007/978-3-642-12538-6_6

[28] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[29] V. Mallawaarachchi, "Introduction to genetic algorithms including example code (2017)," 2019. [Online]. Available: http://www.towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3

[30] Z. K. Akbari, R., "A multilevel evolutionary algorithm for optimizing numerical functions." pp. 419–430, 2011.

[31] Yarpiz. (September 2015) Binary and real-coded genetic algorithms. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/52856-binary-and-real-codedgenetic-algorithms

[32] T. H. Noguchi, N., "Path planning of an agricultural mobile robot by neural network and genetic algorithm," pp. 187–204, 1997.

[33] T. T. Ferentinos, K.P., "Adaptive design optimization of wireless sensor networks using genetic algorithms." pp. 1031–1051, 2007.

[34] S.-s. Y. Li, Y.z., "Application of svm optimized by genetic algorithm in forecasting and management of water consumption used in agriculture." pp. 625–628, 2010.

[35] S. S. Gumaste and A. J. Kadam, "Future weather prediction using genetic algorithm and fft for smart farming," in *2016 International Conference on Computing Communication Control and automation (ICCUBEA)*, Aug 2016, pp. 1–6.

[36] D. E. de Ocampo, A.L.P., "Energy cost optimization in irrigation system of smart farm by using genetic algorithm," pp. 1–7, 2017.

[37] H. U. Hakli, H., "A novel approach for automated land partitioning using genetic algorithm." pp. 10–18, 2017.

[38] K. S. Lee and Z. W. Geem, "A new metaheuristic algorithm for continuous engineering optimization : harmony search theory and practice," 2005, pp. 3902–3933.

[39] A. Askarzadeh and E. Rashedi, *Harmony Search Algorithm*, 03 2017.

[40] N. Rodrigues, "Projeto de controladores pid com meta-heuríisticas de inspiração natural e biológica," 2017.

[41] Z. Geem, "Recent advances in harmony search algorithm." 2010. [Online]. Available: https://doi.org/10.1007/978-3-642-04317-8

[42] Yarpiz, "Harmony search (hs)," September 2015. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/52864-harmony-search-hs

[43] D. C. J. B. A. S. D. Valente, J., "Aerial coverage optimization in precision agriculture management: a musical harmony inspired approach," pp. 153—-159, 2013.

[44] G. A. C. J. C. S. Mandal, S.N., "Prediction of productivity of mustard plant at maturity using harmony search." pp. 933–938, 2012.

[45] N. F. S. P. Brooks and R. King, "Optimization using simulated annealing." vol. 65, pp. 241–257, 1995.

[46] A. E. Van Laarhoven, P.J. (1987) Simulated annealing. in: Simulated annealing: Theory and applications, vol. 37. [Online]. Available: https://doi.org/10.1007/978-94-015-7744-1_2

[47] G. Kendall. (2019) Ai methods simulated annealing. [Online]. Available: http://syllabus.cs.manchester.ac.uk/pgt/2017/COMP60342/lab3/Kendall-simulatedannealing.pdf

[48] H. Fuchigami. (2011) Algoritmo simulated annealing para programa ca o de flow shops paralelos proporcionais com tempo de setup. [Online]. Available: www.din.uem.br/sbpo/sbpo2011/pdf/88031.pdf

[49] Yarpiz, "Simulated annealing (sa)," September 2015. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/52896-simulated-annealing-sa

[50] S.-R. F. L.-J. P. R. H. Pérez-Sánchez, M., "Pats selection towards sustainability in irrigation networks: simulated annealing as a water management too," pp. 234–249, 2018.

[51] C. T.-T. T. Lin, Y.P., "Characterization of soil lead by comparing sequential gaussian simulation, simulated annealing simulation and kriging methods." pp. 189–199, 2001.

[52] R. L.-K. K. Andersen, H.J., "Geometric plant properties by relaxed stereo vision using simulated annealing." 2005.

[53] C. T.-K. T. Brown, P.D., "Optimal on-farm irrigation scheduling with a seasonal water limit using simulated annealing." pp. 892–900, 2010.

[54] N. Rooy. Differential evolution optimization from scratch with python. [Online]. Available: https://nathanrooy.github.io/posts/2017-08-27/simple-differential-evolution-with-python/

[55] S. P. Das, S., "Differential evolution: a survey of the state-of-the-art. ieee trans." 2011.

[56] S. R.-L. J. Price, K. (2005) Differential evolution: A practical approach to global optimization. [Online]. Available: https://doi.org/10.1007/3-540-31306-0

[57] Yarpiz. (2015) Differential evolution (de). [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/52897-differential-evolution-de

[58] O. F. Adeyemo, J., "Differential evolution algorithm for solving multi-objective crop planning model." pp. 848–856, 2010.

[59] B. F.-O. F. Adeyemo, J., "Differential evolution algorithm for crop planning: single and multi-objective optimization model." pp. 1592–1599, 2010.

[60] . C. W. Agrafiotis, D. K., "Feature selection for structure-activity correlation using binary particle swarms," *Journal of Medicinal Chemistry*, vol. 45, pp. 1098–1107, 2002.

[61] T. N. C. Li, S. Yang, "A self-learning particle swarm optimizer for global optimization problems," pp. 627–646, 2012.

[62] R. E. Y. Shi, "Empirical study of particle swarm optimization," pp. 1945–1950, 1999.

[63] P. S. N. Lynn, "Ensemble particle swarm optimizer," vol. 55, pp. 533–548, 2017.

[64] Y. W. H. Liu, Z. Cai, "Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization," vol. 10, pp. 629–640, 2010.

[65] A. E. F. VandenBergh, "A cooperative approach to particle swarm optimization," vol. 8, pp. 225–239, 2004.

[66] J. N. R. Mendes, J. Kennedy, "The fully informed particle swarm: Simpler, maybe better," vol. 8, pp. 204–210, 2004.

[67] P. S. S. B. J.J. Liang, A.K. Qin, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," vol. 10, pp. 281–295, 2006.

[68] G. Wu, "Across neighborhood search for numerical optimization," vol. 329, pp. 597–618, 2016.

[69] J. Kennedy, "The particle swarm: social adaptation of knowledge." pp. 303–308, 1997.

[70] W. S. J. G. Zhang, Y., "A comprehensive survey on particle swarm optimization algorithm and its applications," 2015.

[71] Yarpiz, "Particle swarm optimization (pso)," September 2015. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/52857-particle-swarm-optimizationpso

[72] Z. M. L. G. L. Z. Ji, Y., "Positions research of agriculture vehicle navigation system based on radial basis function neural network and particle swarm optimization." pp. 480–484, 2010.

[73] C. Z. Z. X. Lu, S., "Forecasting agriculture water consumption based on pso and svm." pp. 147–150, 2009.

[74] N. W. Sethanan, K., "Multi-objective particle swarm optimization for mechanical harvester route planning of sugarcane field operations," pp. 969—-984, 2016.

[75] d. M. O. P. C. J. Coelho, J., "Greenhouse air temperature predictive control using the particle swarm optimisation algorithm." pp. 330–344, 2005.

[76] M. P. P. F. Dias, J.A.C., "Privacy-aware ant colony optimization algorithm for real time route planning." pp. 9–9, 2013.

[77] S. T. Dorigo, M., "Ant colony optimization." 2004.

[78] B. Bullnheimer and R. Hartl, "A new rank based version of the ant system: A computational study." *Central European Journal for Operations Research and Economics*, pp. 25–38, 11 1999.

[79] B. M. Dorigo, M., "Ant colony optimization." 2011. [Online]. Available: https://doi.org/10.1007/978-0-387-30164-8

[80] R. Montemanni and D. H. Smith, "Heuristic manipulation, tabu search and frequency assignment," *Comput. Oper. Res.*, vol. 37, no. 3, pp. 543–551, Mar. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.cor.2008.08.006

[81] R. Blum, "A case for school connectedness," *Educational leadership: journal of the Department of Supervision and Curriculum Development, N.E.A*, vol. 62, pp. 16–20, 04 2005.

[82] T. Stützle and H. H. Hoos, "Max-min ant system," *Future Gener. Comput. Syst.*, vol. 16, no. 9, pp. 889–914, Jun. 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=348599.348603

[83] T. Kanade and P. J. Narayanan, "Virtualized reality: Perspectives on 4d digitization of dynamic events," *IEEE Comput. Graph. Appl.*, vol. 27, no. 3, pp. 32–40, May 2007. [Online]. Available: https://doi.org/10.1109/MCG.2007.72

[84] D. Martens, M. De Backer, R. Haesen, J. Vanthienen, M. Snoeck, and B. Baesens, "Classification with ant colony optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 651–665, Oct 2007.

[85] Yarpiz. (September 2015) Ant colony optimization (aco). [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/52859-ant-colony-optimizationaco

[86] A. I. J. M. H. D. G. A. A. Nguyen, D.C.H., "Optimization of irrigation scheduling using ant colony algorithms and an advanced cropping system model." pp. 32–45, 2017.

[87] B. J. V. A. Alaiso, S., "Ant colony optimization for scheduling of agricultural contracting work." 2013.

[88] H. Schnitzler and E. Kalko, "Echolcation by insect eating bats," vol. 51, pp. 557–569, 2001.

[89] A. N. K. Nikov and A. Sahai, "A fuzzy bat clustering method for ergonomic screening of office workplaces," pp. 59–66, 2011.

[90] W. Metzner, "Echolocation behaviour in bats," vol. 75, pp. 453–465, 1991.

[91] X.-S. Yang and A. Gandomi, "Bat algorithm: A novel approach for global engineering optimization," *Engineering Computations*, vol. 29, pp. 464–483, 11 2012.

[92] J. Papa and X.-S. Yang, *Bio-Inspired Computation and Applications in Image Processing*, 07 2016.

[93] X. Yang. (July 2012) Bat algorithm (demo). [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/37582-bat-algorithm-demo

[94] K. S. B. J. Y. X. Senthilnath, J., "A novel approach for multispectral satellite image classification based on the bat algorithm," pp. 599—-603, 2016.

[95] D. F. I. Fister and X. Yang, "A hybrid bat algorithm," 2013.

[96] D. M. S. Saha, R. Kar and S. Ghoshal, "A new design method using opposition-based bat algorithm for iir system identification problem," vol. 5, pp. 99–132, 2013.

[97] A. M. H. Afrabandpey, M. Ghaffari and M. Safayani, "A novel bat algorithm based on chaos for optimization tasks," pp. 1–6, 2014.

[98] G. D. Krishnanand, K., "Detection of multiple source locations using a gloworm metaphor with applications to collective robotics." pp. 84–91, 2005.

[99] F. J. I. Y. X. B. J. Fister, I., "A comprehensive review of firefly algorithms." pp. 34–46, 2013.

[100] X.-S. Yang, "Firefly algorithms for multimodal optimization," pp. 169—-178, 2009. [Online]. Available: https://doi.org/10.1007/978-3-642-04944-6_14

[101] Yarpiz, "Firefly algorithm (fa)," September 2015. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/52900-firefly-algorithm-fa

[102] X.-S. Yang, "Firefly algorithms for multimodal optimization," pp. 169–178, 2009.

[103] I. Y. X.-S. F. I. B. J. Fister, Jr., "Memetic firefly algorithm for combinatorial optimization," 2012.

[104] X.-S. Yang, "Firefly algorithm, levy flights and global optimization," pp. 209—-218, 2012.

[105] A. A. N.-B. M. M. Farahani, S.M., "A gaussian firefly algorithm," p. 448, 2011.

[106] d. A. B. D. M. V. dos Santos Coelho, L., "A chaotic firefly algorithm applied to reliability-redundancy optimization," pp. 517—-521, 2011.

[107] T. M. S. N. Subutic, M., "Parallelization of the firefly algorithm for unconstrained optimization problems," pp. 264—-269, 2012.

[108] Y. X.-S. T. S. A. A. Gandomi, A., "Firefly algorithm with chaos," pp. 89—-98, 2013.

[109] A. S. Gupta, S., "A hybrid firefly algorithm and social spider algorithm for multimodal function," pp. 17—-308, 2016.

[110] S. Kalra and s. Arora, *Firefly Algorithm Hybridized with Flower Pollination Algorithm for Multimodal Functions*, 01 2016, pp. 207–219.

[111] S. Arora, S. Singh, S. Singh, and B. Sharma, "Mutated firefly algorithm," in *2014 International Conference on Parallel, Distributed and Grid Computing*, Dec 2014, pp. 33–38.

[112] B. M. Hosseini, M.S.M., "Optimizing operation of reservoir for agricultural water supply using firefly algorithm." 2014.

[113] W. W.-C. Z. Z. X. Z. J. L. Y. Wang, H., "A new dynamic firefly algorithm for demand estimation of water resources," 2018.

[114] K. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," pp. 52—67, 2002.

[115] A. U.-C. S. Greensmith, J., "Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection." pp. 153–167, 2005. [Online]. Available: https://doi.org/10.1007/11536444_12

[116] N.-P.-H. S. S. Rashedi, E., "Gsa: a gravitational search algorithm," pp. 2232–2248, 2009.

[117] G.-A.-K. E. O. S. R. S. Z. M. Pham, D., "The bees algorithm technical note," 2005.

[118] H. Shah-Hosseini, "Intelligent water drops algorithm: a new optimization method for solving the multiple knapsack problem," pp. 193—212, 2008.

[119] Y.-K. Tamura, K., "Primary study of spiral dynamics inspired optimization," 2011.

[120] D.-S. Yang, X.S., "Cuckoo search via lévy flights," pp. 210–214, 2009.

[121] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.

[122] R. -I. R. F. Rabanal, P., "Using river formation dynamics to design heuristic algorithms," pp. 163—177, 2007. [Online]. Available: https://doi.org/10.1007/978-3-540-73554-0_16

[123] X.-S. Yang, "Flower pollination algorithm for global optimization," pp. 240–249, 2012.

[124] B. A.-O. Z. Eesa, A.S., "Cuttlefish algorithm-a novel bio-inspired optimization algorithm." pp. 1978–1986, 2013.

[125] J. Brownlee, *Clever Algorithms: Nature-Inspired Programming Recipes*, 1st ed. Lulu.com, 2011.

[126] F. F. Orta, A.R. (2018) Aisearch. [Online]. Available: https://aisearch.github.io/

[127] G. W. Xing, B. (2016) Innovative computational intelligence: A rough guide to 134 clever algorithms. isrl, vol. 62. [Online]. Available: https://doi.org/10.1007/978-3-319-03404-1

[128] C. J.-P. E. Oliveira, P.M., "Evolutionary and bio-inspired algorithms in greenhouse control: introduction, review and trends," 2017.