UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

RAFAEL ANDERSON DALMOLIN

CONTROLE SUPERVISÓRIO COOPERATIVO EMPREGANDO A META-HEURÍSTICA SISTEMA DE FORMIGAS

PATO BRANCO

2022

RAFAEL ANDERSON DALMOLIN

CONTROLE SUPERVISÓRIO COOPERATIVO EMPREGANDO A META-HEURÍSTICA SISTEMA DE FORMIGAS

COOPERATIVE SUPERVISORY CONTROL EMPLOYING THE ANT SYSTEM META-HEURISTIC

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. Dr. Marcelo Teixeira
Coorientador: Prof. Dr. Marco Antonio de Castro Barbosa

PATO BRANCO

2022

RAFAEL ANDERSON DALMOLIN ✉

CONTROLE SUPERVISÓRIO COOPERATIVO EMPREGANDO A META-HEURÍSTICA SISTEMA DE FORMIGAS

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de Aprovação: 12 de dezembro de 2022.

Prof. Dr. Marcelo Teixeira
Universidade Tecnológica Federal do Paraná

Profa. Dra. Viviane Dal Molin
Universidade Tecnológica Federal do Paraná

Prof. Dr. Eden Ricardo Dosciatti
Universidade Tecnológica Federal do Paraná

PATO BRANCO

2022

AGRADECIMENTOS

Agradeço aos meus orientadores, por me inspirar a realizar um projeto desse porte, por todo o suporte e apoio que me foi dado durante a elaboração desse trabalho. À todos os meus professores, que me ajudaram a chegar até a conclusão desse trabalho, da escola até a graduação, que me instigaram a ter esse amor pelo conhecimento e estudo.

À toda a minha família, que colaborou direta e indiretamente no meu caminho, à minha mãe e irmão que me ajudaram nessa minha longa jornada pela UTFPR e em especial a minha querida companheira Camila Soares, por ter sempre me incentivado nos momentos difíceis. Aos vários colegas que fiz durante o curso, em especial à aqueles que me acolheram em momentos que precisei e faziam meus dias mais alegres, tenho certeza que serão amizades que irei levar para fora da UTFPR. Além de outras amizades que fiz e que certamente lembrarei com muito carinho.

"Pense na vida que viveu até então como acabada, e como homem morto, veja o que lhe resta como bônus e viva de acordo com a natureza. Ame a mão que o destino lhe deu e a jogue como sua, pois o que haveria de ser mais apropriado?" (MARCUS. . . , s.d., tradução)[1].

[1] "*Think of the life you have lived until now as over and, as a dead man, see what's left as a bonus and live it according to Nature. Love the hand that fate deals you and play it as your own, for what could be more fitting?*" (MARCUS. . . , s.d.).

RESUMO

A indústria moderna contempla um perfil de sistemas que detectam e reagem, em tempo de execução, às mudanças no contexto físico, passando a operar de diferentes modos. Ainda que essas características exponham potencial para agregar melhorias aos processos produtivos, elas também impõem desafios à comunidade de Engenharia de Automação e Controle, como na capacidade limitada para processar em paralelo, múltiplas e complexas malhas de controle. Uma alternativa nasce da combinação de métodos formais de síntese, como a Teoria de Controle Supervisório (TCS), com arquiteturas que descentralizam a implementação do sistema de controle. Isso beneficia tanto a programação do controlador quanto a sua implementação. Entretanto, modelos atuais de implementação não consideram aspectos emergentes da indústria, como controladores inteligentes, por exemplo, que apresentam potencial para utilização eficiente de recursos, personalização da produção e produção cooperativa. Portanto, para alcançar essas competências, este trabalho propõe a construção de um controlador inteligente, a partir da integração de uma meta-heurística de otimização ao sistema de controle advindo da TCS. O espaço de estados resultante do sistema de controle representa todas as possíveis sequências operacionalmente seguras. Assim, a ideia central nesta proposta vem da utilização da meta-heurística Sistema de Formigas (AS), que explora esse espaço construindo cadeias de eventos que, ao serem habilitadas, controlam o comportamento de dispositivos físicos em chão de fábrica. Nessa abordagem, a meta-heurística tem o papel fundamental de construir soluções que criam cooperação de trabalho entre dispositivos e minimizam os custos de produção em função de demandas estocásticas que podem ocorrer. Outro ponto chave nesta proposta vem da utilização de uma arquitetura de controle comunicante, que traz o conceito de um controlador global e diversos agentes locais de controle. O controlador global executa uma abstração do comportamento dos agentes locais, sendo responsável pela coordenação em alto nível, da lógica do sistema de controle, que resulta nas ações físicas locais. Essa abstração permite obter um controlador global mais simples e compacto, reduzindo tanto o espaço de busca quanto a complexidade do problema combinatório e da carga computacional, trazendo benefícios relacionados a performance da meta-heurística implementada. Em termos de resultados, os dados observados trouxeram melhorias aos modelos utilizados e agregaram grande potencial referente à redução de custos e cooperação de trabalho entre dispositivos, por meio da aplicação de métodos meta-heurísticos integrados a um sistema de controle supervisório monolítico, descentralizado ou não. Além disso, demais aspectos de implementação indicam alto potencial de escalabilidade, que implica em possíveis ganhos a TCS, com possibilidade de expansão para diversas pesquisas e uso em ambientes produtivos.

Palavras-chave: Sistema a Eventos Discretos; Teoria de Controle Supervisório; Sistemas de Manufatura; Sistema de Formigas; Problema de escalonamento.

ABSTRACT

The modern industry contemplates a profile of systems that detect and react to – in time for execution – changes in the physical context, operating in different ways. Even though these characteristics show potential to add improvements to the productive processes, they also pose challenges to the Control and Automation Engineering community, such as the limited capacity to process, in parallel, multiple and complex control loops. An alternative appears from the combination of formal methods of synthesis, such as the Supervisory Control Theory (SCT), with architectures that decentralize the implementation of the control system. That benefits both the programming and the implementation of the controller. However, current implementation models don't consider emerging aspects of the industry, such as intelligent controllers, for example, which is something that shows potential for efficient resources usage, production customization and cooperative production. Therefore, to achieve this scope, this work proposes the making of a smart controller from the integration of optimization meta-heuristics to the control system derived from SCT. The state's space from the result of the control system represents all possible sequences that are operationally safe. Thus, the central idea found in this research comes from the use of the mete-heuristic Ant System (AS), which explores this space building chains of events that, when enabled, control the behavior of physical devices on the factory floor. In this approach, the meta-heuristics play a key role in building solutions that create working cooperation between devices and minimizes production costs due to stochastic demands that may occur. Another key aspect of this work comes from the use of a communicating control architecture, which brings the concept of a global controller as well as several local-agents controllers. The global controller performs an abstraction of the behavior of the local agents, being responsible for the high-level coordination of the control system logic, which results in the local physical actions. This abstraction allows the acquirement of a simpler, more compact global controller, reducing the searching as well as the complexity of the combinatorial issue and computational load, bringing benefits related to the performance of the implemented meta-heuristics. In terms of results, the observed data brought improvements to the models used and added great potential regarding cost reduction and work cooperation between devices, through the application of meta-heuristic methods integrated to a monolithic supervisory control system, decentralized or not. in addition, other aspects of implementation indicate high potential for scalability, which implies possible gains for TCS, with the possibility of expansion to various researches and use in productive environments.

Keywords: Discrete Events Systems; Supervisory Control Theory; Manufacturing Systems; Ant System; Job-shop Scheduling.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

SIGLAS

ACO	Otimização por Colônia de Formigas, do Inglês *Ant Colony Optimization*
AF	Autômato Finito, do Inglês *Finite Automaton*
AS	Sistemas de Formigas, do Inglês *Ant System*
CLPs	Controladores Lógicos Programáveis, do Inglês *Programmable Logic Controller*
JSP	Problema de Escalonamento, do Inglês *Job-shop Problem*
PCV	Problema do Caixeiro Viajante , do Inglês *Travel Salesman Problem*
PQA	Problema de Atribuição Quadrática, do Inglês *Problema de atribuição quadrática*
SEDs	Sistemas a Eventos Discretos, do Inglês *Discrete Event System*
SFM	Sistemas Flexíveis de Manufatura, do Inglês *Flexible Manufacturing Systems*
SMs	Sistema de Manufatura, do Inglês *Manufacturing Systems*
TCS	Teoria de Controle Supervisório, do Inglês *Supervisory Control Theory*
UTFPR	Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

O termo *automação* especifica a tecnologia por meio da qual um processo ou procedimento é alcançado sem a assistência humana (ESMAEILIAN; BEHDAD; WANG, B., 2016). Embora se aplique a variados contextos, é na produção industrial que a automação possui maior identidade e se materializa por meio de um conjunto de instruções que é executado por um sistema de controle (REDDY, 2014). Sistemas de Manufatura (SMs) são exemplos clássicos de processos que se utilizam de um elevado grau de automação para transformar matéria-prima em produtos, integrando pessoas, equipamentos e tecnologia (GROOVER, 2011).

As discussões sobre tecnologias de controle e automação industrial se pautavam em aspectos envolvendo robotização, desafios de intertravamentos de comandos e ações, técnicas de implementação em *hardware* de supervisão remota, etc. Recentemente, no entanto, o patamar de discussão evoluiu para uma grandeza que integra tópicos como enxames de robôs coordenados, armazenamento em nuvem, interconectividade, *big data*, cibernética, entre outros (SALDIVAR *et al.*, 2015).

Expostos a esses cenários de inovação, os sistemas legados de controle estão migrando para arquiteturas sensíveis ao contexto, ou seja, sistemas personalizáveis de produção que detectam e reagem a mudanças no contexto do sistema físico, passando assim a operar em múltiplos modos. Esses são denominados *Sistemas Flexíveis de Manufatura* (SFM) (DRATH; HORCH, 2014; YOUSIF, 2016; HACKLER; SIME; WALD, 2015; FERROLHO; CRISOSTOMO, 2007) e possuem como premissa produzir simultaneamente múltiplos tipos de produtos sobre a mesma malha física (ALSZER; KRYSTEK, 2018).

Quando associados a tecnologias computacionais modernas, os SFMs se alinham a métodos avançados para a indústria denominada *Indústria 4.0* (HARRISON; VERA; AHMAD, 2016; BANGEMANN *et al.*, 2016; LIU, Y. *et al.*, 2017; WEN; GUO, 2016; TRAPPEY *et al.*, 2016). No entanto, ainda que a indústria do futuro exponha potencial para agregar melhorias aos processos produtivos, ela também impõe severos desafios à comunidade de Engenharia de Automação e Controle, uma vez que implica no desenvolvimento ou readequação de um amplo conjunto de métodos, técnicas, ferramentas e dispositivos que possam implementar controle sobre processos comunicantes.

Alguns exemplos desses desafios estão ligados a dificuldade de programação lógica de controladores flexíveis (SILVA; RIBEIRO; TEIXEIRA, 2017), aumento da capacidade de memória necessária para embarcá-lo e o maior número de canais de comunicação, isso devido ao

fato de que mapear cada contexto em um sistema resulta em carregar uma lógica de programação particular, cuja codificação manual não é trivial. Isso fica claro em cenários reais, como em linhas de montagem automotivas, em que cada modelo de automóvel requer um processo próprio de fabricação, com características distintas de atuação, que mudam de acordo com o nível de personalização exigido. Como a escolha pelo perfil de atuação depende do controlador, sua programação em geral é bastante complexa ou, em alguns casos, até mesmo inviável de ser obtida via programação manual.

Uma alternativa surge com a adoção de métodos automáticos de síntese de controle e geração de código. Como controladores industriais podem ser observados de acordo com sua evolução a eventos, eles podem ser processados por meio da TCS, proposta por Ramage e Wonham em 1989. A TCS consiste em processar modelos em máquinas de estados para extrair desses modelos um controlador que reúne certas propriedades qualitativas (RAMADGE; WONHAM, W. M., 1989). Embora a TCS seja útil para a obtenção de controladores, ela se mostra limitada para calcular controladores sensíveis ao contexto e sujeito a reconfigurações.

Desse modo, surge o desafio particular deste trabalho, que é estabelecer meios para o suporte inteligente à TCS. Nessa abordagem, o controlador lógico continua com o seu papel elementar de selecionar quais sequências podem ocorrer, mas ele é amparado por uma estrutura a parte que seleciona, dentre os eventos habilitados pelo controlador, quais são mais adequados para habilitar e em que ordem. No geral, a ordem e a sequência dos eventos executados são pré-programados de maneira fixa na memória do controlador lógico e, portanto, sempre são selecionados os eventos pré definidos, não permitindo na prática escalonar ou alternar a ordem e sequência dos eventos.

O problema do escalonamento inteligente tem sido abordado na literatura de inúmeras formas. Umas delas foi utilizando aprendizado por reforço para treinar um agente inteligente que toma decisões em cima dos eventos habilitados pela TCS, tornando o sistema flexível e maximizando o seu desempenho (ZIELINSKI *et al.*, 2021). Outro método bastante aplicado para a resolução desses problemas, tem sido algoritmos heurísticos. Dentre eles pode-se efetivamente destacar Otimização por Colônia de Formigas (ACO), pois este consegue integrar informações específicas e regras do problema (SONG *et al.*, 2007). Algoritmos heurísticos também foram aplicados sobre o espaço de estados resultante da TCS, possibilitando encontrar sequências operacionais que maximizassem o paralelismo no comportamento lógico de um sistema (ALVES; PENA; TAKAHASHI, 2021).

Para a natureza do problema abordado neste trabalho, uma alternativa mais favorável

vem da utilização de algoritmos heurísticos, uma vez que tal assunto pode ser encarado como um problema de otimização, com o nível de complexidade de resolução aumentando exponencialmente conforme o espaço de busca aumenta. Além disso, há também a dificuldade de se obter boas soluções em tempo viável e, portanto, algoritmos meta-heurísticos destacam-se por gerar soluções suficientemente boas para problemas complexos de otimização, quando comparados a outras abordagens que utilizam métodos exatos.

Logo, para o presente trabalho, uma meta-heurística multiagente é estabelecida para tomar decisões sobre o conjunto de eventos habilitados pela TCS. Dessa maneira, a meta-heurística adapta o comportamento do sistema, selecionando sequências de eventos que a nível lógico permitem que os dispositivos cooperem nas suas atividades em prol de objetivos comuns e dinâmicos.

A abordagem utilizada em relação à literatura tem como diferencial o fato de processar a meta-heurística sobre uma abstração de um controlador, trazendo benefícios em aspectos de tempo de processamento e complexidade computacional do problema. Essa abstração não faz parte deste trabalho, sendo desenvolvida em outro projeto paralelo (CANCIO, 2022). Assim, ambos os trabalhos reusam interfaces comuns de resultados. Os resultados alcançados neste trabalho, abrem espaço para inúmeras aplicações industriais, agregando potencial para personalização de produção, uso eficiente de recursos e aspectos de produção cooperativa.

1.1 OBJETIVO GERAL

Este trabalho utiliza um sistema de controle baseado em uma arquitetura de Controle Supervisório Comunicante, desenvolvida em um trabalho complementar (CANCIO, 2022). Tal arquitetura traz o conceito de um controlador centralizado (global) que carrega uma visão abstraída do comportamento dos dispositivos do sistema e se comunica com controladores locais habilitando sequências de eventos operacionalmente seguras, escolhidas a partir de lógicas de controle obtidas via TCS, tanto em nível local, quanto central.

Portanto, aqui é proposto integrar um algoritmo meta-heurístico ao sistema de controle supervisório, para tomar decisões inteligentes baseadas no contexto reportado por cada dispositivo, coordenando as ações dos dispositivos com a intenção de modificar seu comportamento, fazendo-os atuarem cooperando em prol de um objetivo.

1.2 OBJETIVOS ESPECÍFICOS

- Construir um caso prático correspondente a um processo flexível de manufatura, modelá-lo e efetuar a síntese do controlador lógico.
- Implementar o caso prático proposto e o controlador lógico sintetizado.
- Implementar a meta-heurística e integrar ao controlador lógico.
- Simular computacionalmente o modelo proposto e coletar os resultados obtidos da meta-heurística integrada ao controlador lógico, comparando-os com abordagem de controle tradicional.

2 REFERENCIAL TEÓRICO

Os avanços tecnológicos têm tomado dimensões que dificultam a concepção de sistemas informatizados, especialmente aqueles voltados à indústria. O principal obstaculo é a capacidade humana limitada para processar, em paralelo, múltiplas e complexas instâncias de um determinado problema. Essa característica abre, assim, espaço para o desenvolvimento de abordagens formais para a síntese automática de sistemas.

Este capítulo apresenta a fundamentação teórica de uma classe de sistemas que é capaz de tirar proveito de mecanismos formais de modelagem para o processamento e a obtenção automática da estrutura lógica de um sistema, como a sua estrutura de controle automático. Tal fundamentação fornece um aparato robusto para o tratamento de sistemas industriais complexos, sua representação, processamento e eventual otimização.

2.1 SISTEMAS A EVENTOS DISCRETOS

Um sistema é percebido como um agrupamento de inúmeros elementos que inter-relacionam-se com intenção de atingir um objetivo (OGATA, 1996). No mundo real, os sistemas estão presentes em diversas áreas de produção, desde a extração da matéria prima bruta, transformação e processamento da matéria, produção de bens de consumo, entre outras.

De modo geral, os sistemas possuem dois fundamentos básicos, sendo que o estado se relaciona com a identificação do status do sistema em determinado momento, e um procedimento de transição de estados, ditando a forma como seu comportamento atual evolui. Assim, é possível classificar um sistema conforme ele evolui em seu espaço de estados. Nos sistemas contínuos (e.g., sistemas elétricos), por exemplo, as grandezas modeladas evoluem no tempo e, assim, são normalmente representadas por meio de equações diferenciais (OGATA, 1996).

Diferentemente, um sistema pode apresentar grandezas que evoluem em pontos discretos, síncronos ou não no tempo. Quando essa evolução acontece em pontos assíncronos, ela é em geral definida por eventos físicos abruptos e imprevisíveis, dificultando a modelagem por mecanismos de equacionamento. Sistemas que compartilham dessa dinâmica são denominados *Sistemas a Eventos Discretos* (SEDs) (CASSANDRAS; LAFORTUNE, 2009) e a modelagem é, em geral, projetada por meio de diagramas que absorvem e são independentes dessa característica assíncrona no tempo.

Um exemplo simples de um SED pode ser associado a uma máquina que liga e desliga.

Nesse caso, os eventos podem ser pensados como as ações de apertar o botão de ligar e desligar, que são naturalmente assíncronos no tempo. O disparo de um desses eventos força a evolução do sistema para o estado seguinte, em que permanece até a ocorrência de um novo evento.

A família de SEDs é de interesse particular deste trabalho, pois permite modelar intuitivamente sistemas industriais complexos (e.g. robóticos) por meio de diagramas de transição de estados, acompanhar a sua evolução, capturar a parte da sua essência sobre a qual se tem interesse, e processar esses modelos, seja para o propósito de controle, seja para posterior melhoria (otimização) do controlador. Os meios de representação de SEDs são discutidos a seguir.

2.2 MODELAGEM DE SEDS

Uma das maneiras de explorar um sistema é por meio da modelagem, justificando-se por inúmeras causas, uma delas é o fato de permitir verificar com segurança suas principais propriedades, sem a necessidade de atuar experimentalmente sobre sua estrutura real, ou seja, sem interrompê-lo, o que é essencial na indústria.

Dentre os formalismos mais conhecidos para representação dos SEDs estão: as *Redes de Petri* (com suas variadas extensões) (MURATA, 1989), *A teoria das Filas* (BOLCH *et al.*, 2006), a *Teoria dos Autômatos e Linguagens* (HOPCROFT; ULLMAN; MOTWANI, 2001) etc. Para este trabalho, é utilizado a Teoria dos Autômatos e Linguagens como formalismo para modelagem de SEDs, pois mantém seus rigores formais, possuindo uma estrutura adequada à manipulação computacional, e principalmente, porque qualquer operação, sendo ela de caráter exato, deve ser processada sobre um espaço de estados, considerando que os autômatos são estabelecidos no próprio espaço de estados.

2.2.1 AUTÔMATOS FINITOS E LINGUAGENS

A evolução de um SED dá-se por meio da ocorrência de eventos e, como consequência dessa dinâmica, tem-se um percurso formado por sequências de eventos. Para cada uma dessas sequências, dá-se o nome de cadeia. Logo, modelar um SED significa descrever cadeias, e processar esse modelo consiste em manipular cadeias (CASSANDRAS; LAFORTUNE, 2009).

No modelo de um SED, um *alfabeto* é definido como um conjunto composto por todos os eventos possíveis, no qual o conjunto é finito, não-vazio e denotado por Σ. O símbolo Σ^*

denota todas as possíveis combinações finitas de cadeias dos eventos de Σ, incluindo-se a cadeia vazia denotada por ϵ. O comprimento de uma cadeia $s \in \Sigma^*$ é denotada por $|s|$, representando a quantidade de eventos que compõem s, logo $|\epsilon| = 0$, pois não dispõe da ocorrência de nenhum evento (CASSANDRAS; LAFORTUNE, 2009).

Uma linguagem L pode ser definida formalmente como um subconjunto de cadeias em Σ^*, ou seja, $L \in \Sigma^*$, que determina quais possíveis combinações de eventos do Σ são sequências reconhecidas. Por exemplo, seja um alfabeto $\Sigma = \{a,b\}$. Uma linguagem L que contempla todas as cadeias possíveis Σ^* corresponde a $L = \{\epsilon, a,b,aa,bb,ab,ba,aab,...\}$, percebe-se, nesse caso que essa linguagem contempla um número infinito de cadeias, embora cada uma seja finita e derivada do alfabeto também finito (HOPCROFT; MOTWANI; ULLMAN, 1979).

Uma linguagem pode ser classificada como regular ou não regular, sendo a regular a classe que pode ser expressa por um *Autômato Finito* (AF) (SIPSER, 2012), também conhecido como máquina de estados finitos, que é definido como uma 5-tupla $\langle Q,\Sigma, \rightarrow, q_0, F\rangle$, consistindo em:

- Q é o conjunto finito de estados.
- Σ é o alfabeto finito.
- $\rightarrow \subseteq Q \times \Sigma \times Q$ é a relação de transição de estados.
- $q_0 \in Q$ é o estado inicial.
- $F \subseteq Q$ é o subconjunto de estados marcados.

A função de transição $\rightarrow$ especifica as regras de movimentação, ou seja, ao estar em um estado q_0, a partir de um símbolo de entrada α, o próximo estado a ser alcançado será q_1. Essas regras podem ser denotadas matematicamente por $q_0 \xrightarrow{\sigma} q_1$ (SIPSER, 2012).

Uma maneira de representar graficamente um autômato, é por meio de um dígrafo, ou seja, um grafo dirigido. Este, facilita sua interpretação e manipulação, e assim, resguarda sua integridade formal. Um exemplo de autômato é ilustrado na Figura 1.

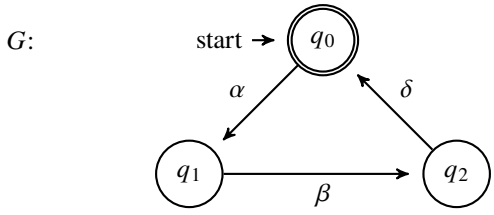


Figura 1 – Exemplo de autômato
Fonte: Autoria própria

Os nós do grafo representam os estados do autômato, as arestas identificam as transições, denotadas por setas direcionadas interligando os estados correspondentes. Assim, as arestas são associadas com os respectivos eventos que desencadeiam uma transição de estados. O estado inicial é demarcado por uma seta uni-conectada e os estados com círculos duplos definem um estado marcado ou estado de aceitação. De modo geral, os estados marcados estão associados à ideia de tarefa completa (CASSANDRAS; LAFORTUNE, 2009).

Dois tipos de linguagens podem ser definidas a partir de um autômato qualquer G, a linguagem gerada e a linguagem marcada, definidas respectivamente como:

$$L(G) = \{s \in \Sigma^* : q_0 \xrightarrow{s} q \in Q\};$$
$$L^w(G) = \{s \in \Sigma^* : q_0 \xrightarrow{s} q \in F\}.$$

A linguagem gerada $L(G)$ abrange todas as possíveis cadeias do autômato G, ao passo que a linguagem marcada $L^w(G)$ representa todo o conjunto de cadeias que atingem os estados marcados, tal que $L^w(G) \subseteq L(G)$ (CASSANDRAS; LAFORTUNE, 2009).

2.2.2 OPERAÇÕES SOBRE AUTÔMATOS E LINGUAGENS

Uma vez que as linguagens podem ser definidas como conjuntos, todas as suas operações tradicionais podem ser aplicadas, como a união e a intersecção. Além dessas, outras operações específicas podem ser definidas sobre uma linguagem (CURY, 2001; TEIXEIRA, 2013), exemplos dessas operações serão apresentadas a seguir.

a. *Operações morfológicas*: a partir de um alfabeto Σ e uma cadeia $s = xyz$, contendo $x,y,z \in \Sigma^*$, define-se que:

   - x é um dos prefixos de s;
   - y é uma das subcadeias de s;
   - z é um dos sufixo de s.

b. *Concatenação*: sejam duas linguagens L_1, $L_2 \subseteq \Sigma^*$, a concatenação de L_1 em L_2, define-se como:

$$L_1L_2 = \{s \in \Sigma^* : (s = s_1s_2) \wedge (s_1 \in L_1 \wedge s_2 \in L_2)\}.$$

c. *Prefixo-fechamento*: dado uma linguagem $L \subseteq \Sigma^*$, o prefixo-fechamento, denotado por $\bar{L}$, define-se como:

$$\bar{L} = \{s \in \Sigma^* : (\exists t \in \Sigma^*), st \in L\}.$$

Ou seja, a linguagem $\bar{L}$ contém todos os prefixos de todas as cadeias de L. Se L for prefixo-fechada, logo $L = \bar{L}$, e qualquer prefixo da cadeia de L é também uma cadeia de L.

Outra operação útil que pode ser aplicada em linguagens e autômatos é denotado por ||, e dá-se o nome de composição síncrona (WONHAM, W., 2002). Neste trabalho, por questões de praticidade, define-se somente a operação de sincronização para autômatos, pois este é adotado como formalismo para simplificar a modelagem de SEDs.

Sejam dois autômatos, $G_1 = \langle Q_1,\Sigma_1, \rightarrow_1, q_1^0, F_1\rangle$ e $G_2 = \langle Q_2,\Sigma_2, \rightarrow_2, q_2^0, F_2\rangle$, representado na Figura 2.

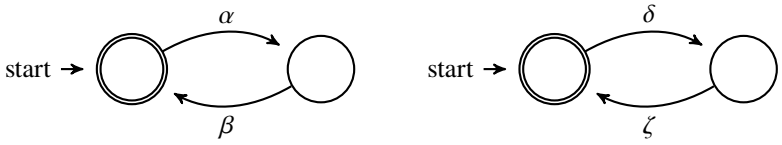


Figura 2 – Modelagem dos autômatos
Fonte: Autoria própria

A composição síncrona de G_1 e G_2, denotada por $G_1||G_2$, é definida pelo seguinte autômato $G = G_1||G_2 = \langle Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, \rightarrow_1 || \rightarrow_2, (q_1^0,q_2^0), F_1 \times F_2\rangle$, tal que suas transições são definidas pelas regras da Eq. (1).

$$\rightarrow_1 || \rightarrow_2 = \begin{cases} (q_1,q_2) \xrightarrow{\sigma} (q_1',q_2')) & \text{se} \quad \sigma \in \Sigma_1 \cap \Sigma_2; \\ (q_1,q_2) \xrightarrow{\sigma} (q_1',q_2)) & \text{se} \quad \sigma \in \Sigma_1 \backslash \Sigma_2; \\ (q_1,q_2) \xrightarrow{\sigma} (q_1,q_2')) & \text{se} \quad \sigma \in \Sigma_2 \backslash \Sigma_1; \\ \text{Indefinido caso contrário} \end{cases} \tag{1}$$

De outro modo, a operação || permite que eventos comuns a ambos autômatos possam ser executados de maneira síncrona, ao passo que demais eventos ocorram assincronamente, isto é, de maneira independente para apenas um autômato (CURY, 2001). O resultado da composição pode ser observado na Figura 3.

Assim como para dois autômatos, a composição síncrona pode ser estendida a n autômatos, obtendo um modelo global G. Então, sejam os autômatos $G^i = \langle Q_i,\Sigma_i, \rightarrow_i, q_i^0, F_i\rangle$ para $i = 1,...,n$, o modelo pode ser obtido por meio da seguinte expressão:

$G = G1 \,||\, G2$:

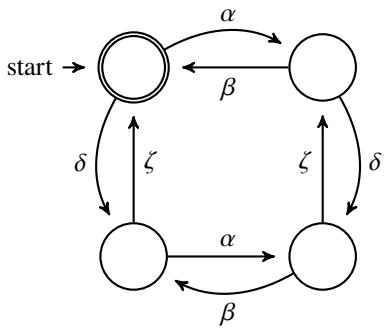


Figura 3 – Composição síncrona de autômatos
Fonte: Autoria própria

$$G = \prod_{i=1}^{n} G^i, \text{ tal que } \Sigma = \bigcup_{i=1}^{n} \Sigma_i.$$

Enquanto o comportamento gerado e marcado resultantes da composição entre n autômatos, como mostrados por (WONHAM, W., 2002), são definidos abaixo:

$$L(G) = \prod_{i=1}^{n} L(G^i) \text{ e } L^w(G) = \prod_{i=1}^{n} L^w(G^i).$$

Assim, percebe-se que uma vantagem da utilização da composição síncrona, é permitir lidar individualmente com modelos mais simples G^i. Compondo-os na sequência para a formação de um modelo único global G, também conhecida como planta (CASSANDRAS; LAFORTUNE, 2009). Este modelo representa o comportamento do sistema em malha aberta, pois não sofre interferência externa, ou seja, não é submetida a nenhuma ação de controle.

2.3 CONTROLE DE SEDS

Uma vez modelada a planta em malha aberta, surge a necessidade de que o sistema cumpra um determinado comportamento. Para isso, são implementadas um conjunto de restrições, denominadas especificações e denotadas por E. Assim como a planta, as especificações também podem ser representadas por autômatos, desse modo, é possível realizar a modelagem das especificações por meio de vários autômatos E^i, cada um restringindo o comportamento local de um componente, aplicando a composição síncrona obtém-se uma especificação geral do sistema, denotada por meio da seguinte expressão:

$$E = \prod_{i=1}^{n} E^i$$

Para o autômato da Figura 3, considera-se que se queira restringir o acontecimento de alguns eventos, o que pode ser realizado pela especificação da Figura 4. Esta é designada a proibir que o evento δ ocorra no estado inicial, ou seja, o evento δ só pode ocorrer após um evento β. Outra restrição é imposta sobre o evento α, que só pode acontecer após o evento δ.

Um fato importante ao se modelar uma especificação está ligado à marcação dos estados, em geral o papel de definir se uma tarefa é completa ou não, é incumbência da planta. Portanto, uma opção conveniente é, sempre manter a marcação de todos os estados na especificação.

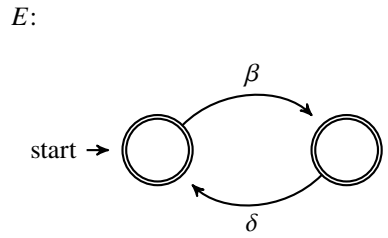


Figura 4 – Modelagem da especificação
Fonte: Autoria própria

Por fim, quando realizado a composição da planta com a especificação, obtém-se $K = E \parallel G$. Representado na Figura 5, correspondendo ao comportamento desejado da planta sob efeito da ação de controle da especificação, denominado modelo de malha fechada, tal que, $L^w(K) \subseteq L^w(G)$.

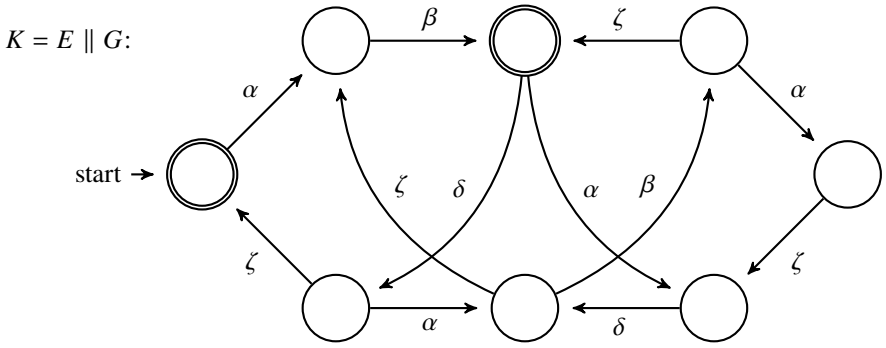


Figura 5 – Comportamento em de malha fechada
Fonte: Autoria própria

Seria intuitivo tomar a decisão de se aproveitar dessa composição para a implementação de um controlador, utilizando-o para desabilitar dinamicamente os eventos de G. No entanto, eventos como o fim de operação de uma máquina ou a quebra de um equipamento, possuem a particularidade de serem espontâneos e independentes de qualquer política de controle. Isto é, não podem ser diretamente desabilitados, e caso não sejam considerados, causam um problema de controlabilidade. Além disso, deseja-se que os controladores possuam a propriedade de não-bloqueio.

A ideia de não bloqueio refere-se à capacidade do sistema de atingir um estado marcado, isto é, completar uma tarefa. De modo formal, um autômato $G = \langle Q_G, \Sigma_G, \rightarrow_G, q_G^0, F_G \rangle$ é dito não bloqueante se cada cadeia de $a \in L(G)$ é prefixo de uma cadeia marcada. Em outras palavras, se cada cadeia $a \in L(G)$ pode ser completada por algum $w \in \Sigma^*$ tal que $aw \in L^w(G)$. Ou ainda, se $L(G) = \overline{L^w(G)}$.

Logo, faz-se necessário estabelecer um método de diferenciar a natureza dos eventos que integram o sistema, levando em consideração as propriedades requeridas para permitir ou não, sua habilitação na planta. Assim, a TCS (RAMADGE; WONHAM, W. M., 1989) incorpora a controlabilidade em um SED ao definir uma estrutura responsável pelo particionamento do conjunto de eventos da planta, por $\Sigma = \Sigma_c \cup \Sigma_u$, tal que Σ_c é o conjunto de eventos controláveis, cuja ocorrência não é involuntária e pode ser evitada na planta, e Σ_u denota o conjunto de todos os eventos não-controláveis, os quais não podem ser diretamente desabilitados.

Então, uma linguagem $L(K)$ é dita controlável em relação a planta, caso respeite a seguinte expressão:

$$\overline{L(K)}\Sigma_u \cap L(G) \subseteq \overline{L(K)}.$$

De outro modo, após qualquer prefixo de $L(K)$, se um evento não-controlável é verificado em $L(G)$, ainda tem-se como resultado uma cadeia como prefixo de $L(K)$. Este fato, garante que nenhum evento não-controlável possível de acontecer em $L(G)$, esteja sendo desabilitado pela ação de controle.

Isso significa que se o sistema atinge um estado q, a partir do qual um evento incontrolável a é possível, e a é desabilitado pela especificação, então q é um mau estado, porque viola o princípio da controlabilidade. Por fim, q é removido, assim como todas as transições associadas a este estado.

A partir disso, a TCS calcula a operação denominada síntese (CASSANDRAS; LAFORTUNE, 2009), que resulta em uma entidade, esta, efetivamente, implementa as ações de controle na planta e é denominada *supervisor S*, denotada sup$C(K,G)$, na qual representa a máxima linguagem controlável, ou a sub-linguagem controlável que mais se aproxima de $L(K)$. Desse modo, $supC(K,G)$ pode ser associado ao comportamento menos restritivo possível de ser implementado por um supervisor não-bloqueante S sobre G, de maneira a respeitar o conjunto de especificações.

O algoritmo que calcula o sup$C(K,G)$ sobre os autômatos G e E pode ser sumarizado pelos três passos a seguir:

- Passo 1: identificar maus estados em E. Caso não existam então $S = E$, chega-se ao fim.
- Passo 2: remover os maus estados em E. Caso existam, deve-se atualizar E, eliminando os maus estados.
- Passo 3: testar o bloqueio em E e retorna ao passo 1.

Seja, então, o exemplo da Figura 5, representando o sistema em malha fechada, sobre efeito da ação de controle modelada na especificação da Figura 4, assim, é possível observar que o modelo K, contém claramente um comportamento não-bloqueante. Porém, ao assumir que os eventos de término β e ζ, são eventos não-controláveis, verifica-se que existem estados indesejáveis em K, portanto, este modelo não respeitará a propriedade de controlabilidade.

Desse modo, ao se utilizar da TCS, obtém-se um supervisor não-bloqueante, controlável e minimamente restritivo S, ilustrado na Figura 6, em que os maus estados e transições a si relacionadas, são removidos. Esses maus estados estão demarcados com um x na cor vermelha.

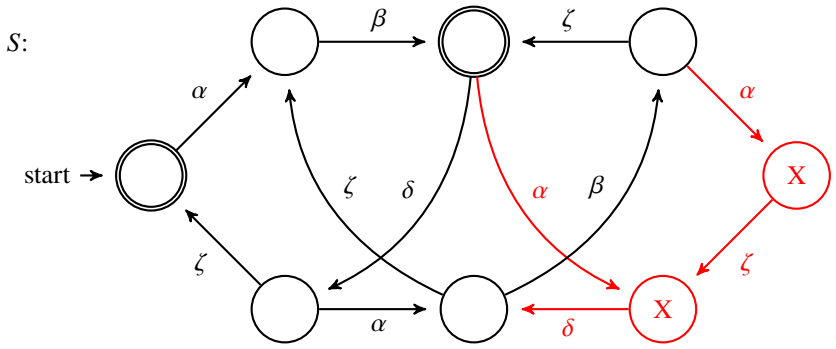


Figura 6 – Comportamento do supervisor
Fonte: Autoria própria

2.4 CONTROLE DESCENTRALIZADO

Uma vez que o supervisor foi sintetizado, é possível implementá-lo por meio de uma conversão da sua máquina de estados em código em alguma linguagem de programação. Sua implementação pode ser efetuada de vários modos, uma delas em um único *Hardware* de maneira centralizada (e.g., Controladores Lógicos Programáveis (CLPs) (LEAL; DA CRUZ; HOUNSELL, 2009). Essa, especificamente, com a desvantagem de ter uma lógica fixa e pouca flexibilidade. Outra forma, de modo descentralizado, em que o controlador pode ser implementado em vários dispositivos (AFZALIAN; NOORBAKHSH; NABAVI, 2008).

A ideia de implementação descentralizada está muito alinhada a cooperação de sistemas

e aos moldes da indústria (*indústria 4.0*), logo, quando se opta por descentralizar uma arquitetura de controle, os dispositivos tornam-se mais efetivos, configurados localmente, e possibilitando maior flexibilidade, o que resulta em sistemas distribuídos (HARRISON; VERA; AHMAD, 2016).

Assim, surge a necessidade de que esses subsistemas troquem informações e sejam coordenados. Para a coordenação assume-se que as partes menores do processo se coordenam localmente, além de reportarem informações a uma estrutura global, responsável por controlar os dispositivos locais. Esse tema é explorado seguidamente, no entanto, a comunicação não é abordada, pois não faz parte do escopo deste trabalho.

2.5 CONTROLE GLOBAL

O fato de descentralizar uma estrutura de controle, traz consigo a possibilidade de ter um agente global coordenando vários dispositivos. Por exemplo, se antes uma grande fábrica seria controlada por meio de um CLP, agora, na concepção descentralizada, é possível gerenciar inúmeras fábricas em lugares distintos do mundo, cada qual controlada por meio de um agente comandado pelo controlador global. Este centraliza todas as decisões, de modo que todos os dispositivos atuam de maneira passiva aguardando um comando para começar a operar. Em contrapartida, é esperado que o controlador global possua uma visão abstrata dos pormenores dos controladores locais.

Assim, se um agente central consegue atuar como um controlador global de uma estrutura toda modular, então é conveniente que esse agente consiga tomar decisões inteligentes, pois estas influenciam diretamente no comportamento dos dispositivos passivos. Então, é intuitivo pensar que, nessa estrutura modular, os dispositivos passem a atuar cooperando um com o outro, de modo a cumprir uma determinada tarefa ou objetivo.

2.6 COOPERAÇÃO DE AGENTES

O problema de tomar decisões inteligentes em um sistema industrial possui grande importância, uma vez que sua resolução impacta de diferentes maneiras em cima de uma linha de produção. Dentre esses impactos, os mais relevantes são a utilização eficiente dos recursos disponíveis, minimização do Makespan (Período de tempo desde o início e término de um processo) e throughput (Quantidade de produtos entregues durante um período).

A chave para encontrar eficiência dá-se por meio de um planejamento da produção, utilizando técnicas de escalonamento de tarefas, ou seja, escolher boas sequências operacionais (WANG, W.; YUAN; LIU, X., 2008). Assim, a escolha inteligente dessas sequências pode levar os dispositivos a cooperarem entre si em função de um objetivo, permitindo algum grau de otimização de produção.

Em uma implementação de controle convencional, o que se deseja, é encontrar uma boa sequência de eventos que retratam o funcionamento desejado do sistema, assim essa sequência é implementada em um CLP. Porém, nesse modo de implementação não existe um escalonamento de eventos, portanto as ações realizadas sob controle do CLP são meramente repetitivas e nenhum dispositivo coopera.

Uma maneira de resolver esse problema é implementar um algoritmo que atua em cima do espaço de estados do controlador global e combina as possíveis sequências, determinando a melhor de acordo com uma função objetivo. Entretanto, encontrar a melhor sequência entre todas as sequências disponíveis, que representam o comportamento desejado, por exemplo em termos de Makespan, acaba compondo um problema não polinomial (GAREY; JOHNSON, 1979), que torna os problemas de escalonamento intratáveis.

Para lidar com essa classe de problemas, os métodos heurísticos estão entre as abordagens mais comuns encontradas na literatura (ALMEDER; MÖNCH, 2011; SANTOS; DOURADO, 1999; PENA *et al.*, 2016). Os algoritmos heurísticos buscam melhorar a eficiência do processo de encontrar soluções. Eles não garantem encontrar a solução ótima para um problema, mas são muito eficientes em retornar uma solução de qualidade em tempo apropriado (polinomial).

Isso significa que alguns tipos de algoritmos podem ser empregados para solucionar esse problema (e.g., Algoritmos gulosos), que são algoritmos construtivos, isto é, tem vantagem por sua fácil implementação, rápida execução e, na teoria, podem até fornecer a melhor solução (caso ideal). Por outro lado, esse tipo de abordagem nem sempre conduz a soluções ótimas globais, pois sempre escolhe o evento localmente ótimo em cada fase.

Com o avanço constante da tecnologia, a indústria tem disponível um maior poder computacional. Desse modo, permite aos pesquisadores da área abordar esse problema por meio de algoritmos mais complexos (e.g., Meta-heurísticas de refinamento) que demandam maior poder de processamento, mas, em contrapartida, tem maiores probabilidades de retornar soluções melhores (GOLDBARG, E.; GOLDBARG, M.; LUNA, 2017).

Na literatura, meta-heurísticas são utilizadas com maior frequência para solução de problemas de otimização combinatória, as mais comuns são algoritmos genéticos, algoritmos

meméticos, colônia de formigas e enxame de partículas (TALBI, 2009). Esse problema de otimização combinatória pode ser facilmente observado na tentativa de encontrar a melhor sequência operacional no espaço de estados do controlador global, e que reflita na cooperação dos dispositivos em prol de um objetivo.

No entanto, escolher o algoritmo mais adequado a ser empregado fica a cargo do projetista, pois este depende da dinâmica do problema, da sua modelagem e da função objetivo pretendida. Na próxima seção, será apresentada uma meta-heurística de refinamento e multiagente. Neste trabalho, a meta-heurística será integrada ao controlador global, utilizando uma abstração do espaço de estados do supervisor sintetizado via TCS e, será responsável por instituir, de maneira inteligente, a cooperação entre os dispositivos.

2.7 SISTEMA DE FORMIGAS

O algoritmo base utilizado para instituir a cooperação dos dispositivos, neste trabalho, será fundamentado no *Ant System* (Sistema de Formigas), que é uma meta-heurística populacional de otimização proposta por (DORIGO; MANIEZZO; COLORNI, 1996). O algoritmo é inspirado no comportamento forrageiro das formigas, que se movimentam de maneira organizada, com objetivo de encontrar alimento e retornar ao formigueiro. Esse comportamento, é um dos problemas estudados por etólogos, que visam entender como animais quase cegos, como as formigas, conseguem criar caminhos mais curtos dos seus ninhos até fontes de alimentos.

No mundo real, as formigas vagam aleatoriamente até encontrar o seu alimento e, durante esse percurso excretam no solo uma substância química denominada feromônio. Nesse contexto, descobriu-se que os indivíduos utilizavam-se de trilhas de feromônios para transmitir informações sobre seus caminhos, fazendo com que outras formigas que encontrassem essa trilha, utilizem a informação proveniente do feromônio para decidir sua rota. Assim, as trilhas com maiores quantidades de feromônios depósito, consequentemente passam a ter maiores probabilidades de serem escolhidas (DORIGO; GAMBARDELLA, L., 1997).

Mediante ao exposto, novas formigas que passam a seguir um determinado caminho, também acabam reforçando a trilha com seu próprio feromônio. Desse modo, surge um comportamento coletivo autocatalítico, em que, quanto mais formigas seguem o caminho, mais atrativo esse caminho se torna para formigas posteriores seguirem (DORIGO; MANIEZZO; COLORNI, 1996). Porém, com o passar do tempo, os feromônios começam a evaporar reduzindo sua força atrativa. Seguindo essa lógica, quanto mais tempo levar para uma formiga percorrer um caminho,

mais os feromônios evaporam.

Assim, a tendência é dos caminhos curtos serem percorridos com maior frequência, tornando-os a ter uma maior densidade de feromônios em comparação aos caminhos longos. A proposta do algoritmo Sistema de Formigas é imitar esse comportamento, criando um processo de construção de solução com formigas artificiais, que ao percorrer um caminho, excretam uma determinada quantidade de feromônios que irá influenciar a decisão das formigas posteriores.

A abordagem proposta no Sistema de Formigas, foi utilizada para solucionar o clássico Problema do Caixeiro Viajante (PCV). Além disso, devido ao fato da sua versatilidade e robustez, também é possível de ser aplicado na resolução de outros problemas de otimização combinatória (e.g.,problema de atribuição quadrática (PQA), *job-shop problem* (JSP) e similares.) (DORIGO; MANIEZZO; COLORNI, 1996).

Nesse sentido, estima-se que sua utilização agregue melhorias para estabelecer um suporte inteligente, selecionando sequências operacionais possíveis, dentro do espaço de estados do controlador global. Para expor a implementação e funcionamento do algoritmo, aborda-se sua aplicação no contexto do PCV, consistindo em encontrar o menor circuito para visitar um conjunto de cidades que estão completamente conectadas entre si, passando apenas uma vez por cada cidade.

O PCV pode ser modelado por um grafo ponderado não direcionado, de tal maneira que, as cidades sejam identificadas nos nodos ou (vértices) do grafo e os caminhos que interligam as cidades sejam as arestas. Assim, cada caminho que interliga duas cidades ou mais estão associadas a custos, que por sua vez, correspondem as distâncias entre as cidades, representado como informação heurística da aresta (DORIGO; DI CARO; GAMBARDELLA, L. M., 1999). A implementação do Sistema de Formigas ocorre em três etapas:

(i) Primeira etapa: Uma quantidade de formigas artificiais são alocadas em vértices distintos ou iguais do grafo, a cada passo, uma formiga se desloca no grafo construindo uma solução estocasticamente, ou seja, a solução representa a ordem em que as arestas do grafo que serão percorridas.

(ii) Segunda etapa: Os caminhos encontrados por cada formiga são comparados e a melhor solução global é guardada.

(iii) Terceira etapa: Consiste em atualizar os níveis de feromônios presentes em cada aresta (DORIGO; MANIEZZO; COLORNI, 1996).

O funcionamento do Sistema de Formigas pode ser observado por meio do pseudo

código ilustrado na Figura 7.

Algorithm 1 Sistema de formigas

```
Inicializa parâmetros;
while condição de parada não for atingida do
    while existe formiga sem caminho do
        Gerar uma solução;
    end
    Atualizar solução global;
    Atualizar feromônios;
end
Retorna solução global;
```

Figura 7 – Pseudo código Sistema de Formigas
Fonte: (DORIGO; MANIEZZO; COLORNI, 1996)

Na primeira etapa, para a formiga construir uma solução, é necessário selecionar qual aresta irá percorrer para transitar dentro do grafo, a Figura 8 ilustra uma possível situação. Uma formiga alocada em uma cidade inicial tem nesse momento 3 possibilidades de caminhos que a levam a cidades distintas. Então, a partir da sua posição atual, a formiga considera a informação heurística (distância) e nível de feromônio disponível em cada aresta para, então, calcular a probabilidade de transição.

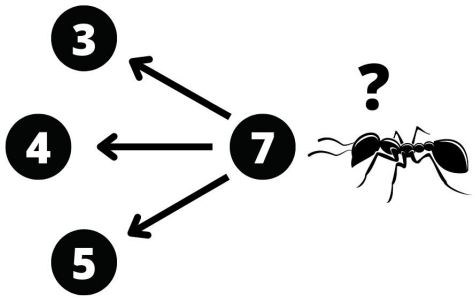


Figura 8 – Decisão de transição de cidades
Fonte: Autoria própria

A probabilidade de escolha da aresta a ser transitada dentro de um conjunto de arestas disponíveis, é definida pela Eq. (2). Também para não haver a possibilidade de uma formiga escolher cidades já visitadas, o Sistema de Formigas tem memória, que pode ser representada por meio de uma lista tabu (lista de proibições), não permitindo visitar uma cidade duas vezes enquanto constrói sua solução (DORIGO; GAMBARDELLA, L., 1997).

Para melhor entendimento, a Eq.(2) expressa a probabilidade da formiga k transitar da cidade x para y durante a iteração t. O cálculo depende da combinação dos valores τ e η, que

respectivamente representa a trilha de feromônio, indicando o quão proficiente foi transitar por essa aresta no passado e a informação heurística.

Os parâmetros α e β controlam a influência dos feromônios e da informação heurística e N_x^k é o conjunto de cidades vizinhas da cidade x ainda não visitadas pela formiga k (DORIGO; MANIEZZO; COLORNI, 1996).

$$p_{x,y}^k(t) = \left\{ \frac{[\tau_{x,y}(t)]^\alpha \cdot [\eta_{x,y}(t)]^\beta}{\sum N_x^k [\tau_{x,y}(t)]^\alpha \cdot [\eta_{x,y}(t)]^\beta} \right. \tag{2}$$

Diante disso, passo a passo, uma formiga se desloca sobre o grafo aplicando uma decisão estocástica baseada na informação heurística e densidade de feromônios, até encontrar um caminho, gerando uma solução. Após todas as formigas construírem uma solução, ocorre a segunda etapa, atualizando a melhor solução encontrada, como solução global e, posteriormente, a última etapa é executada, consistindo no processo de atualização dos feromônios envolvendo tanto o depósito quanto a evaporação.

A evaporação dos feromônios ao longo do tempo conduzem as formigas a seguirem outros caminhos, possivelmente podem ser caminhos piores, mas são importantes para evitar a convergência para uma solução localmente ótima (DORIGO; BIRATTARI *et al.*, 2004). A evaporação tem importância pelo fato de atuar diversificando os caminhos escolhidos, caso não existisse evaporação, os caminhos escolhidos pelas primeiras formigas tenderiam a ser excessivamente mais atrativos para as formigas seguintes. Ou seja, isso tornaria a exploração do espaço de solução restrita. A atualização do feromônio é definida pela Eq. (3) a seguir:

$$\tau_{x,y} = (1 - \rho) \cdot \tau_{x,y} + \sum_{k=1}^{m} \Delta\tau_{x,y}^k \tag{3}$$

O parâmetro ρ regula a taxa de evaporação do feromônio, m indica o número de formigas e $\Delta\tau_{x,y}^k$ é definido por:

$$\Delta\tau_{x,y}^k = \begin{cases} \frac{Q}{L^k}, & \text{Se aresta xy faz parte do caminho da formiga k} \\ 0, & \text{Caso contrário} \end{cases} \tag{4}$$

O parâmetro L^k representa o custo da rota percorrida pela formiga k, já Q assume a quantidade de feromônio excretada por uma formiga. Nessa última etapa, apenas a formiga que gerou a melhor solução daquela iteração tem permissão para depositar seus feromônios sobre sua trilha. Assim, esse processo se repete por várias iterações até encontrar a condição de parada, sendo esta definida de acordo com as características do problema (DORIGO; MANIEZZO;

COLORNI, 1996).

Nesse sistema, nota-se que o algoritmo é uma heurística de refinamento, pois durante a primeira iteração constrói uma solução inicial baseada estocasticamente nos feromônios e informação heurística. Cada iteração vai refinando a solução por meio de um feedback positivo deixado por cada trilha de feromônio das iterações anteriores.

Observa-se que há vários parâmetros que devem ser ajustados para se obter um bom desempenho. Portanto, é fundamental o ajuste correto de α e β para se obter um bom equilíbrio entre a diversificação e intensificação. Para isso, os parâmetros α e β devem ser maiores que 0, e também, valores muito altos de α em relação a β, fazem as formigas escolherem majoritariamente arestas escolhidas por formigas anteriores, causando uma estagnação precoce do algoritmo. Caso contrário, e β seja muito maior em relação à α, as escolhas se comportaram de maneira semelhante a uma escolha estocástica gulosa (DORIGO; MANIEZZO; COLORNI, 1996).

O parâmetro ρ que regula a taxa de evaporação, deve ser ajustado para um valor menor que 1, evitando que ocorra o acúmulo ilimitado de feromônios. A quantidade de formigas geralmente é definida igual a quantidade de cidades do problema. Por fim, é proposto que inicialize-se os feromônios em todas as arestas com $1/n.Lnn$, em que Lnn é o custo de construir uma solução puramente gulosa (DORIGO; MANIEZZO; COLORNI, 1996).

No próximo capítulo, será visto como é possível integrar o Sistema de Formigas no controlador global, para atribuir cooperação entre dispositivos em busca de cumprir objetivos predefinidos.

3 CONTROLE SUPERVISÓRIO COOPERATIVO EMPREGANDO A META-HEURÍSTICA SISTEMA DE FORMIGAS

Este capítulo apresenta uma proposta para integrar um algoritmo meta-heurístico ao controle supervisório de SEDs. O controle em questão é baseado em uma arquitetura de controle supervisório comunicante, desenvolvida em trabalho complementar (CANCIO, 2022). Tal arquitetura utiliza o conceito de um controlador centralizado (global), que observa a planta e se comunica com controladores locais habilitando sequências de eventos operacionalmente seguras, escolhidas a partir de lógicas de controle obtidas via TCS, tanto em nível local, quanto central.

Na arquitetura utilizada, os dispositivos em chão de fábrica são controlados localmente, seguindo o conceito de descentralização discutido na seção 2.4. Isto é, suas ações são implementadas localmente em *hardware* dedicado e observam o contexto local na íntegra, porém, não observam o contexto geral da planta, o que está a cargo do controle central. Assim, a descentralização força que os dispositivos locais recebam e enviem informações para o controlador central, com intuito de confirmar as ações a serem executada localmente, bem como para atualizar o controlador global sobre os status da execução local, abrindo, desse modo, margem para políticas de cooperação entre agentes.

O foco deste trabalho é, portanto, desenvolver uma política de cooperação para controladores descentralizados, usando a meta-heurística Sistema de Formigas. Será mostrado que é possível que controladores a eventos, caracterizados por serem exatos, tomem decisões inteligentes e adaptem o seu comportamento para atuar de maneira cooperativa em prol de demandas em comum.

Um exemplo prático de um sistema flexível de manufatura é explorado para ilustrar o funcionamento da arquitetura com controlador global. Porém, seu funcionamento não se limita somente a esse caso, uma vez que a versatilidade da meta-heurística permite sua aplicação em versões análogas do exemplo abordado, com mínimo retrabalho. Estruturalmente, o trabalho está organizado conforme as etapas mostradas na Figura 9.

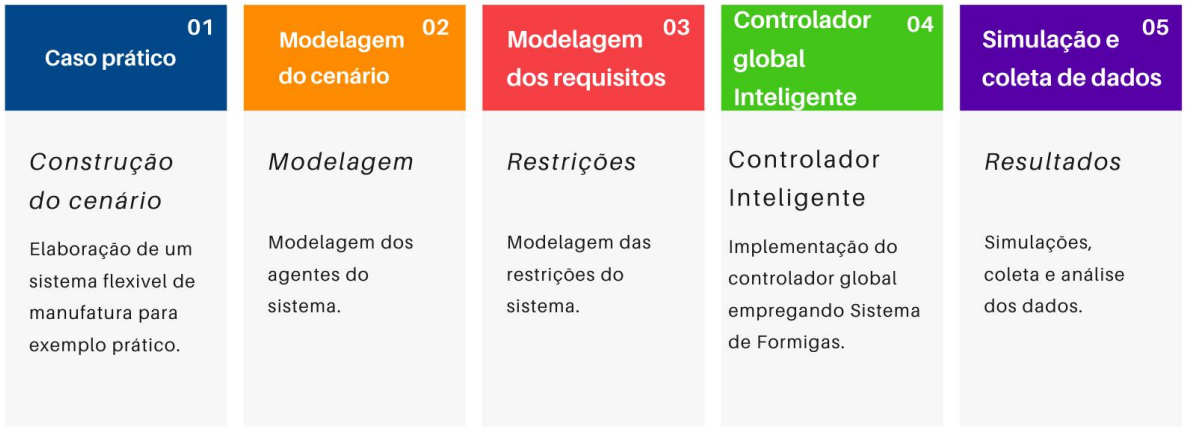


Figura 9 – Exemplo de imagem das etapas de organização
Fonte: Autoria própria

3.1 EXEMPLO PRÁTICO EXPLORADO

O exemplo prático abordado neste trabalho ilustra um contexto industrial de um sistema de manufatura voltado à fabricação de engrenagens, contando com dois dispositivos robóticos denominados de agentes, que disputam por recursos com o intuito de produzir dois tipos de engrenagens, com características e qualidades distintas.

O cenário do exemplo prático pode ser observado na Figura 10, sendo dividido em dois tipos de regiões: particulares e compartilhadas, respectivamente demarcadas em preto e vermelho. Enquanto as regiões particulares são destinadas somente ao tráfego do agente a qual ela pertence, regiões compartilhadas, por sua vez, podem ser acessadas por qualquer um dos agentes, desde que estejam disponíveis. Além disso, essas regiões são subdivididas em outras quatro áreas, nomeadas de neutra, manutenção, pré-processamento e finalização.

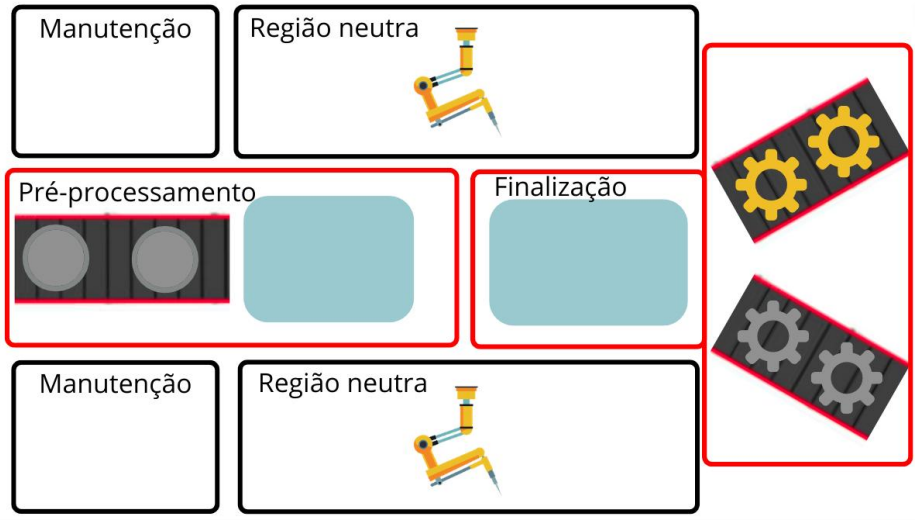


Figura 10 – Cenário elaborado para implementação do controlador inteligente
Fonte: Autoria própria

O funcionamento do sistema se dá da seguinte maneira: inicialmente os agentes estão alocados em suas regiões particulares, mais especificamente em suas áreas neutras, que são zonas de segurança, sendo que neste local é possível trafegar seguramente sem eventuais colisões. Então, a partir desta área, os agentes têm total liberdade para se deslocar a qualquer uma das outras três áreas do esquema e, assim, iniciar suas atividades de manufatura.

A área de pré-processamento refere-se à primeira etapa de produção. Nesta, há disponíveis ferramentas e máquinas para realização de recortes e furos. Portanto, para a manufatura de uma engrenagem, o agente deve retirar um lingote de metal bruto de um *buffer*, que contém alimentação infinita de matéria-prima, para então efetuar nessa área uma série de ações, que posteriormente definirão se a engrenagem será de um modelo A ou B.

Já a área de acabamento é o local onde a engrenagem será transportada pelos agentes advindos da etapa de preparação, realizando assim a segunda etapa de produção. Nessa área, os agentes executam ações para remover potenciais saliências das arestas, têmpera do material para que se atinja um nível de endurecimento necessário, efetuando também o descarte em sua respectiva esteira. Por fim, a última área do esquema destina-se à manutenção do agente.

Outra particularidade importante desse sistema é que nas regiões compartilhadas entre ambos os agentes haverá a possibilidade de colisão, sendo de responsabilidade do sistema de controle atuar para evitá-las. Consequentemente, somente um agente por vez poderá acessar uma área compartilhada, exigindo, assim, que os agentes cooperem para a devida utilização desses locais.

A partir do sistema proposto é possível, por meio de SEDs — vide o que foi retratado na seção 2.2 —, definir a modelagem que abstrai os principais comportamentos de cada agente, bem como o comportamento esperado para atuação em conjunto. Assim, é possível calcular um supervisor compacto a partir de TCS, conforme elucidado na seção 2.3, que será utilizado para implementar o controlador global, obtendo resultados compatíveis com as especificações a serem definidas, evitando problemas de controlabilidade e bloqueios, e atuando de forma minimamente restritiva.

3.2 MODELAGEM DO SISTEMA DE CONTROLE

Esta seção traz informações sobre as etapas dois e três, relacionadas a obtenção do sistema de controle implementado no controlador, iniciando desde a modelagem dos dispositivos e suas restrições e concluindo até o cálculo do supervisor. Por fim, é exposto a aplicação de uma

abstração na captura dos eventos, trazendo benefícios para calcular um supervisor compacto.

A modelagem do sistema foi realizada utilizando a ferramenta de *software* Supremica, que disponibiliza um ambiente gráfico de modelagem, simulação, síntese de um supervisor via TCS e verificação de SEDs. Assim, é assegurado a obtenção de um supervisor com características de controlabilidade, minimamente restritivo e não-bloqueio. Por conseguinte, tal supervisor pode ser representado por uma máquina de estados.

A modelagem das ações disponíveis no agente 1 para a realização das suas atividades de manufatura são ilustradas pela máquina de estados denominada G_1, que pode ser vista na Figura 11. Igualmente, a máquina de estados referente às ações disponíveis no agente 2, é denominado por G_2 e exposta na Figura 12.

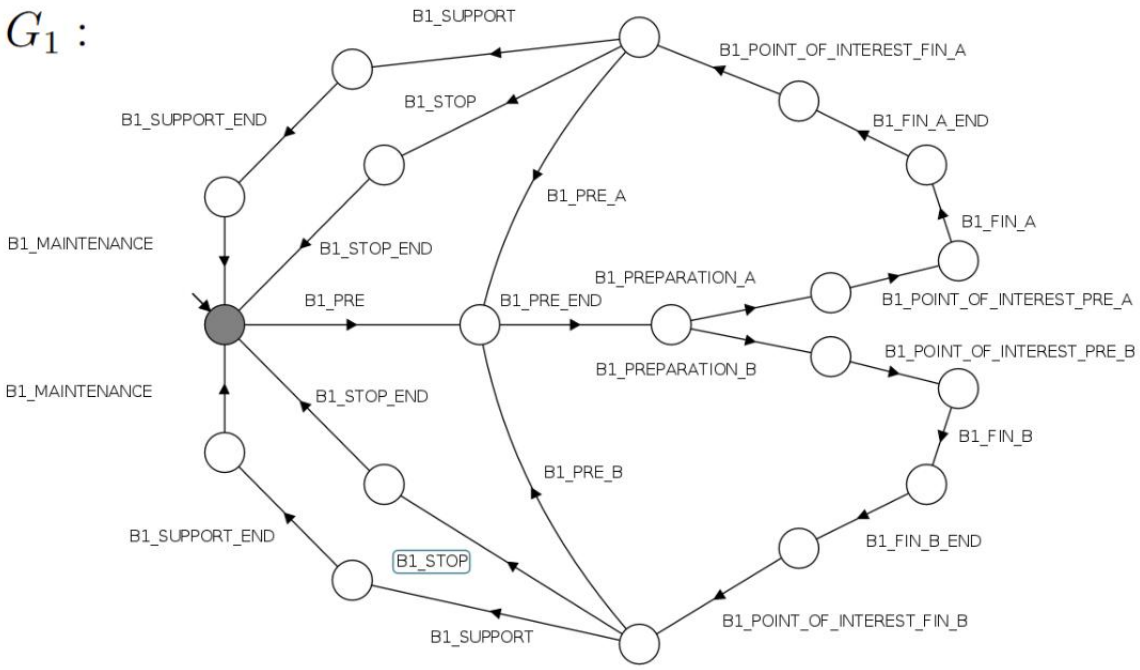


Figura 11 – Autômato do agente 1
Fonte: Autoria própria

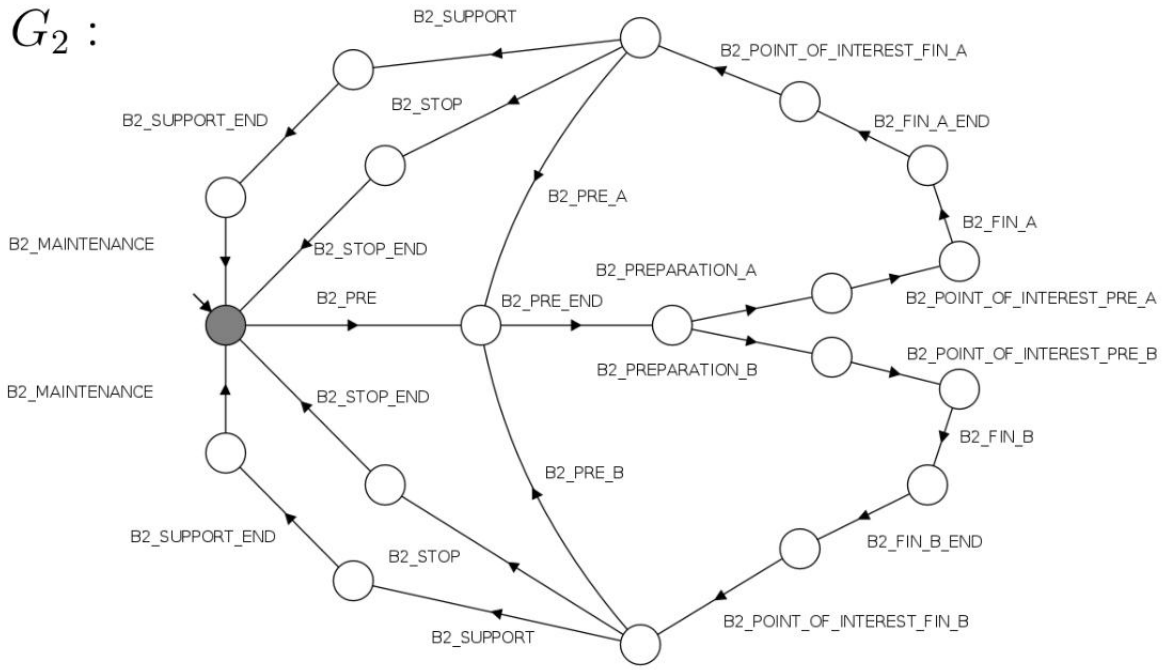


Figura 12 – Autômato do agente 2
Fonte: Autoria própria

Vale ressaltar que o agente 2 executa eventos equivalentes ao agente 1, ou seja, um evento B1_PRE do autômato G_1 existirá no autômato G_2 como o evento B2_PRE, contendo apenas alterações do prefixo B1 para B2 nos rótulos dos eventos. Dessa forma, para esclarecer o funcionamento dos autômatos da Figura 11 e 12, utiliza-se o contexto do cenário projetado na Figura 10, abordando os eventos capturados na modelagem do autômato G_1.

Inicialmente, o agente 1 está alocado em sua região neutra e, para manufatura de uma engrenagem, deve se direcionar à área de preparação. Esse movimento é capturado por dois eventos no autômato G_1: B1_PRE que indica o sinal de início do movimento e B1_PRE_END de término, sinalizando que o dispositivo já se encontra na área de preparação. Os próximos eventos disponíveis no autômato G_1 habilitam a opção de escolher o modelo de engrenagem a ser manufaturado, sendo representado por B1_PREPARATION_A e B1_PREPARATION_B.

Respectivamente, os eventos subsequentes aos eventos anteriores são B1_POINT_OF_INTEREST_PRE_A e B1_POINT_OF_INTEREST_PRE_B, e representam uma abstração de pormenores eventos de preparação do produto. Tais eventos são explicados de maneira mais aprofundada na seção 3.2.1. Esses eventos representam fisicamente etapas nas quais o agente realiza ações de preparo, cada ação de acordo com o modelo de produto escolhido.

Após o término da etapa de preparo, o próximo evento dará sequência a manufatura do produto e, para isso, o produto deve ser transportado para área de finalização. Ou seja, fisicamente

os eventos de início B1_FIN_A ou B1_FIN_B fazem com que o agente se mova com o produto semi preparado para a área de finalização.

Deste modo, os eventos de término B1_FIN_A_END e B1_FIN_B_END indicam que o agente chegou com sucesso naquela região. Consequentemente, os eventos subsequentes B1_POINT_OF_INTEREST_FIN_A ou B1_POINT_OF_INTEREST_FIN_B, também representam uma versão abstraída dos eventos que devem ser realizados para o acabamento, assim como o descarte final do produto até sua respectiva esteira.

Portanto, a partir desse momento, o autômato G_1 habilita eventos para iniciar uma transição para a área de produção inicial por meio de B1_PRE_A; ir para uma região neutra desocupando a área de acabamento optando por B1_STOP; ou, por fim, ir para a área de realização de manutenção escolhendo B1_SUPPORT. Assim, o ciclo se repete tanto para a produção de uma engrenagem do modelo A ou B.

As Tabelas 3 e 5 no apêndice trazem os rótulos dos eventos capturados durante o processo de modelagem dos autômatos G_1 e G_2; a descrição semântica dos eventos; características de controlabilidade e, ainda, o custo relativo de optar por aquele evento. Esse custo, posteriormente, é importante para a construção do algoritmo integrado ao controlador global.

Após concebidas as modelagens dos autômatos G_1 e G_2, é possível realizar a operação composição, resultando no autômato denominado G, que, por si só, não possui nenhuma ação de controle. Para tal autômato, foram modeladas as especificações E_1 e E_2, ilustradas na Figura 13 e 14. Tais especificações representam o conjunto de regras que os agentes devem seguir para acessar os recursos das regiões compartilhadas, impondo assim restrições exclusivamente aos eventos da planta G.

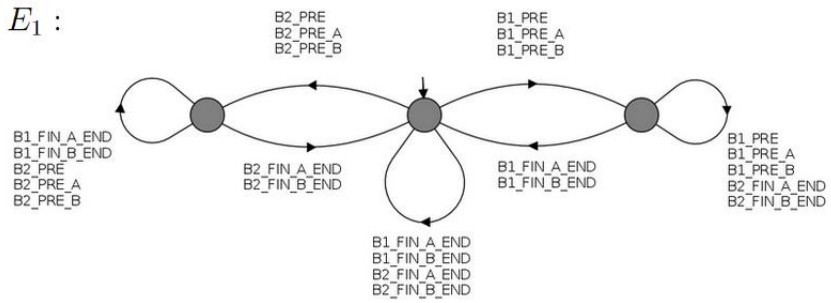


Figura 13 – Restrições para acessar a região de preparação
Fonte: Autoria própria

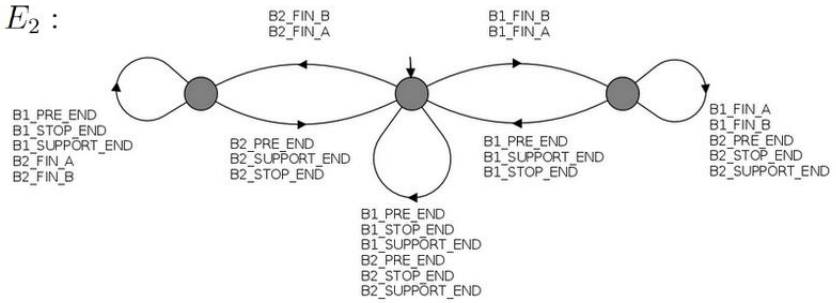


Figura 14 – Restrições para acessar a região de finalização
Fonte: Autoria própria

As especificações e suas respectivas finalidades sobre a planta G são descritas a seguir:

- E1: Essa especificação possui a responsabilidade de efetuar um *mutex* na região de preparação exposta na Figura 10. Isto é, se um dos agentes acessar essa região, o outro ficará proibido de entrar até que o primeiro saia.
- E2: Igualmente a E_1, essa especificação também implementa um *mutex*, porém, na região de finalização exposta na Figura 10. Desse modo, a especificação E_2 também permite que somente um agente possa adentrar nesta região, proibindo o outro de entrar até que o primeiro saia.

Em posse do autômato G e suas respectivas especificações E_1 e E_2, um supervisor S_{Abs} é calculado por meio do *software* Supremica, resultando em uma máquina de estados contendo 207 estados, com 434 eventos de transição. O supervisor obtido, representa o sistema de controle obtido via TCS, e que, em conjunto com uma meta-heurística apresentada mais adiante, atuará como um controlador global do sistema.

A abordagem para a síntese do supervisor utilizada via TCS foi a monolítica, no entanto, mesmo que seja a maneira mais básica de se obter um sistema de controle via TCS, gerou-se ganhos com um supervisor mais compacto. Outro pormenor a ser destacado é que os autômatos utilizados até aqui foram construídos apenas para o cálculo de S_{Abs}, diferentemente dos quais realmente são implementados localmente nos *hardwares* dos agentes.

A seguir é exposto como os eventos abstraídos favorecem para calcular o supervisor S_{Abs} de forma compacta. Além de uma comparação com um supervisor sem abstração, para fins de validar este argumento.

3.2.1 ABSTRAÇÃO DOS EVENTOS

O supervisor S_{Abs} obtido na seção 3.2 carrega consigo alguns eventos denominados de abstraídos. Um evento abstraído representa ações mapeadas por um conjunto de outros eventos. Por exemplo, caso um conjunto de eventos mapeados dentro de um processo não afetem as propriedades de controlabilidade e segurança da TCS, tal conjunto será reduzido apenas ao mapeamento do início e término desse processo.

A Figura 15 ilustra esse comportamento de abstração, com o autômato A_1 expondo o mapeando de um processo qualquer. Tal processo também é mapeado pelo autômato A_2, porém, com uma visão abstraída em dois eventos.

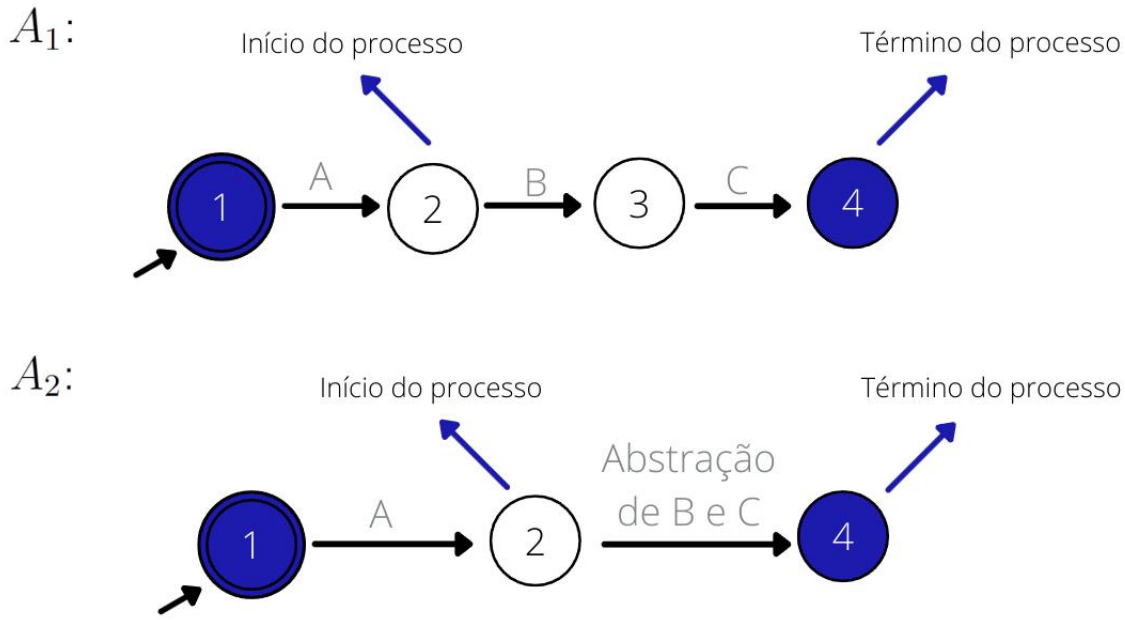


Figura 15 – Mapeamento abstraído de um processo
Fonte: Autoria própria

Uma metodologia para ditar quais eventos devem ser abstraídos ou mantidos na modelagem dos autômatos não é explorada por este trabalho, mas é citada como uma possibilidade para outros trabalhos complementares. O processo de abstração de eventos neste trabalho foi realizado observando o contexto do exemplo prático explorado. Diante disso, os eventos que reúnem características de abstração para o cálculo do supervisor S_{Abs} são:

- B1_POINT_OF_INTEREST_PRE_A;
- B1_POINT_OF_INTEREST_PRE_B;
- B1_POINT_OF_INTEREST_FIN_A;
- B1_POINT_OF_INTEREST_FIN_B;

- B1_MAINTENANCE.

Mediante ao exposto, para cada evento abstraído dentro do supervisor S_{Abs}, existe associado um conjunto de outros eventos que estão implementados localmente como rotina no hardware dos agentes. Consequentemente, os autômatos G_1 e G_2 utilizados no cálculo do supervisor S_{Abs} carregam alguns eventos que são abstrações de um conjunto de eventos que realmente estão implementados localmente nos autômatos dos agentes.

A abstração utilizada nos eventos é possível graças a descentralização do controle, uma vez que os agentes implementam localmente uma máquina de estados com todos os eventos desse conjunto em seu *hardware*. Sendo assim, é possível mapear apenas os eventos iniciais e finais de um conjunto, não sendo necessário observar o contexto dos eventos intermediários de um processo.

Em outras palavras, a abstração utilizada implica que se o supervisor habilitar um evento de início de uma abstração, o conjunto de eventos abstraído também é habilitado como rotina nos agentes. Logo, quando o último evento do conjunto porventura ocorrer, o evento de término associado no supervisor também ocorrerá.

Para expor evidências da vantagem obtida ao calcular um supervisor utilizando-se da abstração dos eventos, outro supervisor S_{NotAbs} foi sintetizado utilizando os autômatos implementados localmente pelos agentes 1 e 2. A Figura 16 ilustra o autômato $G_{1_{NotAbs}}$, que é similar $G_{2_{NotAbs}}$, contando apenas com mudanças no prefixo dos eventos.

Estes autômatos são modelados sem abstração em dois eventos. O evento B1_ABS_A representa outro possível movimento de trabalho durante a etapa de preparação de uma engrenagem do tipo A. Respectivamente, B1_ABS_B representa um movimento idêntico, porém, utilizado na manufatura de uma engrenagem do tipo B.

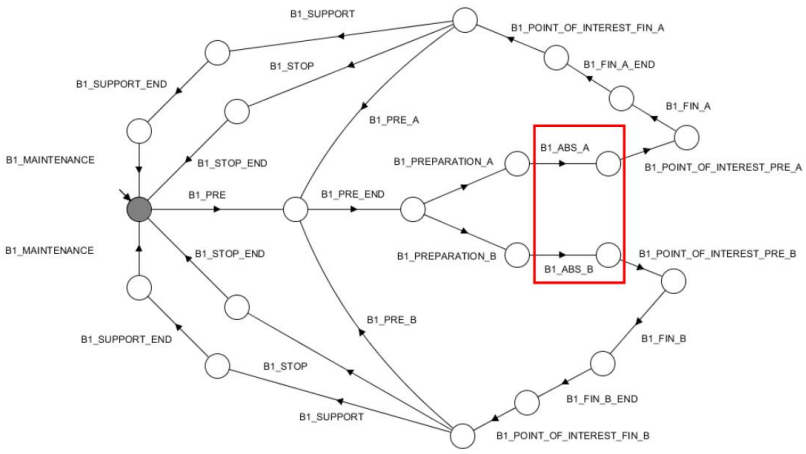


Figura 16 – Autômato $G_{1_{NotAbs}}$ com possíveis ações do agente 1
Fonte: Autoria própria

Assim, se os autômatos $G_{1NotAbs}$ e $G_{2NotAbs}$ forem compostos e posteriormente sintetizados junto às especificações $E1$ e $E2$, obtém-se um sistema de controle também monolítico S_{NotAbs}, que equivale ao controle do sistema S_{Abs}, não fazendo uso, entretanto, do conceito de abstração em todos seus eventos. A comparação da quantidade de estados e transições obtidas no S_{NotAbs} podem ser observados na Tabela 1.

Supervisor	**Estados**	**Transições**
S_{Abs}	207	434
S_{NotAbs}	251	526

Tabela 1 – Supervisores sintetizados com e sem o processo de abstração de eventos.
Fonte: Autoria própria

A abstração dos eventos possibilitaram reduzir a complexidade do espaço de estados do supervisor S_{Abs}. Nesta simples comparação, observou-se uma redução de 44 estados e 92 transições, o que são ganhos consideráveis, visto que apenas 4 eventos foram deixados de ser abstraídos no supervisor S_{NotAbs}. É importante ressaltar que esses ganhos podem ser variáveis, a depender da quantidade de eventos que existem no conjunto a ser abstraído.

Uma vez que o algoritmo meta-heurístico utiliza o espaço de estados do supervisor S_{Abs}, este, por ser mais compacto, agrega visivelmente benefícios no que se refere à capacidade de processamento computacional necessária e, na complexidade do problema combinatório para encontrar sequências de produção, pois a complexidade de calcular uma sequência de eventos suficientemente ótima e em tempo viável aumenta exponencialmente conforme o tamanho do espaço de estados se expande.

Na próxima seção será discutida a integração da meta-heurística Sistema de Formigas ao supervisor S_{Abs}, referente a etapa 4 deste trabalho, permitindo construir um controlador inteligente.

3.3 CONTROLADOR GLOBAL COM SISTEMA DE FORMIGAS

O sistema de controle obtido na seção 3.2 garante a segurança e a robustez das ações, algo que é elementar em processos de fabricação. Além disso, os processos de fabricação visam ainda alcançar algum nível de otimização para uso eficiente de recursos, tempo de produção, custo, entre outros aspectos.

Todavia, essa otimização demandada não é possível de ser alcançada somente com a utilização da TCS, pois quando se trata de adicionar flexibilidade ao sistema de controle a TCS se torna um tanto quanto limitada, por se pautar num modelo exato de acionamento. Assim, em

casos de mudanças de prioridades e alterações no processo, é necessário uma remodelagem do projeto, remapeando cada contexto para sintetizar um novo supervisor, tornando a programação do sistema bastante complexa ou, em outros casos, inviável.

Dessa maneira, afim de adicionar características de flexibilidade ao sistema de controle, um controlador inteligente é proposto a partir da integração da meta-heurística Sistema de Formigas, com o espaço de estados advindo do supervisor projetado pela TCS, que preserva sequências de eventos operacionalmente seguras. Na Figura 17 pode ser visto a arquitetura sob qual o controlador atua.

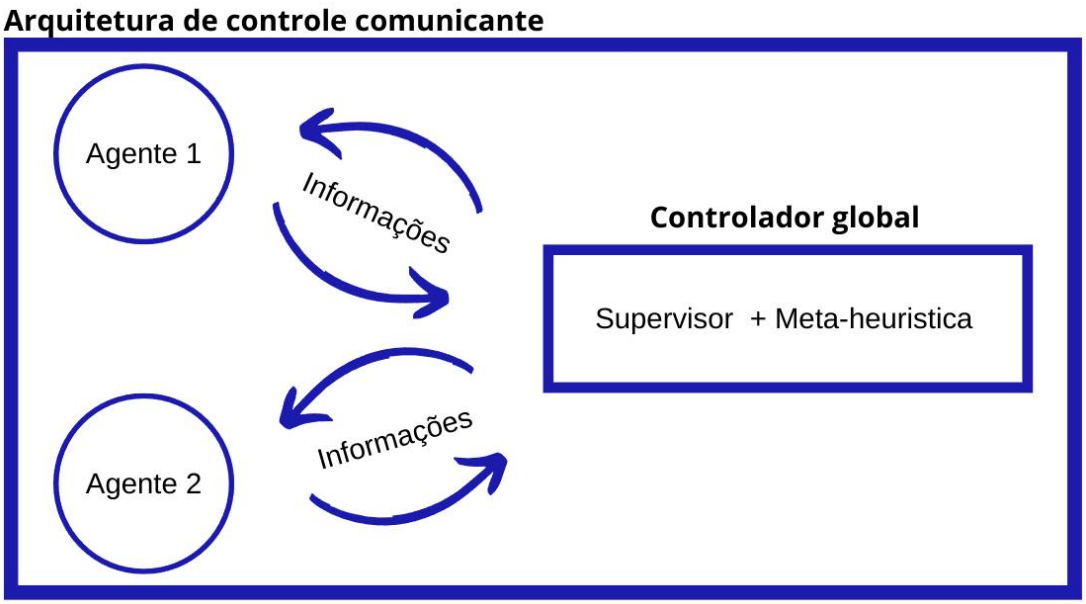


Figura 17 – Arquitetura de controle comunicante
Fonte: Autoria própria

Esta arquitetura implementa um sistema de comunicação descentralizado via internet (CANCIO, 2022). Seu funcionamento se dá da seguinte maneira: os agentes, contém implementados em seu *hardware* uma máquina de estados com todas suas ações físicas, e sem nenhum evento abstraído. Por outro lado, o controlador global pode ou não carregar uma máquina de estados com uma visão abstraída dos processos, realizando o controle conforme programado em seu *hardware*.

Portanto, para desenvolvimento deste trabalho, assume-se que os agentes conseguem trocar informações com o controlador global solicitando quais eventos podem ser habilitados ou desabilitados, para atualizar o controlador global sobre seu contexto atual de trabalho.

Assim, a partir da arquitetura apresentada, este trabalho foca na construção de um controlador inteligente para atuar como controlador global. A implementação do controlador inteligente faz uso do Sistema de Formigas, que tem a incumbência de explorar o espaço de

estados do supervisor S_{Abs}, planejando as ações dos agentes a fim de cumprir uma determinada demanda de engrenagens A e B. Dessa maneira, o algoritmo consegue gerar um conjunto de eventos que se habilitados sequencialmente, adapta o comportamento dos agentes, criando cooperação de trabalho e reduzindo custos de produção.

O fato de utilizar a arquitetura de comunicação via internet agrega a possibilidade de sintetizar um sistema de controle com espaço de estados mais compacto. Além disso, é importante lembrar que o Sistema de Formigas também pode ser implementado em outros controladores, sem utilizar-se desta arquitetura, porém apresentando desvantagem em ter que explorar um espaço de estados muito maior.

3.3.1 RESTRIÇÕES APLICADAS AO SISTEMA DE FORMIGAS

Para implementação do controlador inteligente, algumas restrições criadas e aplicadas aos agentes em determinadas circunstâncias precisam ser semanticamente capturadas e representadas por meio de outros métodos dentro do Sistema de Formigas. Por exemplo, limitar a quantidade de ações permitidas de trabalho por agente, evitando que acumulem um nível de desgaste acima do permitido para prevenir sua quebra.

Esse aspecto relacionado ao desgaste devido ao esforço de trabalho, trouxe a necessidade de adotar um coeficiente de desgaste, sendo fundamental para induzir o aumento da probabilidade de enviar os agentes para a manutenção preventiva. Assim, esse coeficiente se altera de forma dinâmica de acordo com o contexto do ambiente, ou seja, quanto mais os agentes se aproximam do acúmulo máximo de desgaste, maiores as chances do controlador enviá-los para área de manutenção.

Também foi definido que, ao acumular um desgaste por manufaturar 5 produtos, o agente precisa se dirigir a área de manutenção. Além do coeficiente de desgaste, uma lista tabu (lista de movimentos proibidos) foi introduzida para bloquear eventos inconvenientes, isto é, eventos que levam a ultrapassar o nível máximo de desgaste, bem como eventos que levam os agentes a manufaturar mais produtos que a demanda requisitada.

Tais restrições incluídas interferem diretamente na forma de calcular a probabilidade de habilitar um determinado evento. Logo, se a probabilidade a ser calculada for de um evento de manutenção, o coeficiente de desgaste é levado em consideração no cálculo. Do contrário, se for um evento pertencente a lista tabu, tal evento é desconsiderado, sendo impossível de ser habilitado pelo controlador global. Por fim, a probabilidade de todos os outros eventos, são

calculadas conforme a Eq. (2).

Definidas essas restrições, a função objetivo adotada para o Sistema de Formigas foca em encontrar o conjunto de eventos que façam os agentes cooperarem para realizar a manufatura dos produtos com mínimo custo possível. Outro ponto importante é fazer o controlador sempre dar prioridade por habilitar sequências de eventos que permitam o paralelismo dos agentes.

3.3.2 AJUSTE DO ESPAÇO DE ESTADOS DO SUPERVISOR

O Sistema de Formigas é uma técnica probabilística que visa resolver problemas computacionais que podem ser modelados por meio de grafos ponderados. Então, a máquina de estados do supervisor S_{Abs} pode ser vista como espaço de busca do algoritmo. Desta forma, cada transição pode representar um evento a ser habilitado para ocorrer dentro dos sistema de controle dos agentes.

Assim sendo, cada evento do supervisor S_{Abs} semanticamente representa uma ação física que o agente pode realizar em chão de fábrica e pode ser relacionado a um custo monetário para sua execução. Feito essa relação, o algoritmo pode percorrer o espaço de estados considerando os custos dos eventos, visando encontrar um conjunto de eventos que satisfaçam a função objetivo e todas as restrições discutidas na seção 3.3.1.

Para construir um grafo ponderado a partir do supervisor S_{Abs}, todos os eventos foram atrelados a um custo monetário, os controláveis por se tratarem de eventos de início de um processo, receberam um custo um tanto quanto baixo (simbólico), induzindo ao algoritmo aumentar as chances de escolher habilitar eventos de início de processo, incitando sempre fazer os agentes trabalharem em paralelo.

Por outro lado, os eventos de término de um processo ficaram com o custo total de executar a ação daquele processo em si. Já os eventos que representam a abstração de pormenores partes receberam um custo igual a soma de todos os eventos do conjunto abstraído. As Tabelas 3 e 5 trazem todos os eventos do sistema, suas respectivas descrições semânticas, tal como características e relações entre os custos de execução.

Em resumo, até esta etapa do trabalho as discussões foram pautadas, principalmente, em questões conceituais relacionadas à implementação do sistema de controle do exemplo prático e construção de um controlador inteligente baseado em Sistema de Formigas. A seguir, serão apresentadas as tecnologias e ferramentas utilizadas para realizar a implementação do controlador e executar as simulações, testes e coleta de dados.

3.3.3 TECNOLOGIAS E FERRAMENTAS PARA IMPLEMENTAÇÃO

Para o desenvolvimento deste trabalho, um conjunto de ferramentas e tecnologias foram escolhidas levando em consideração a aplicação do controlador em diferentes ambientes, tanto de desenvolvimento quanto produtivo, além da possibilidade de integração do sistema de controle inteligente em outros projetos.

Para as etapas dois e três referentes ao sistema de controle, foi utilizado o *software* Supremica (AKESSON *et al.*, 2019), pois este possibilita, a partir de um ambiente gráfico, modelar, simular e sintetizar um supervisor via TCS. Além disso, permite exportar o sistema de controle para um arquivo com linguagem de marcação *Extensible Markup Language* (XML), capaz de descrever diversos tipos de dados. Logo, essas informações podem ser facilmente manipuladas e interpretadas por uma linguagem de programação.

A etapa de implementação do controlador inteligente, foi desenvolvida utilizando a ferramenta web Jupyter Notebook (JUPYTER. . . , 2022), que auxilia na compilação de códigos de diferentes linguagens de programação. Já a linguagem de programação utilizada foi Python (PYTHON. . . , 2022), que é uma linguagem de programação de alto nível e, reúni um grande número de recursos e bibliotecas para manipulação e visualização de dados, aumentando muito a relevância de sua escolha para as etapas de simulações, testes e coletas de dados.

Outra característica importante na utilização da linguagem Python é sua compatibilidade com a Arquitetura de Controle Comunicante, desenvolvida em outro projeto (CANCIO, 2022). Essa escolha eventualmente possibilita também a incorporação do controlador inteligente em outros trabalhos complementares, como a criação de uma interface para entrada de demandas de produção, ou visualização de informações em tempo real dos status de trabalho dos dispositivos.

Vale ressaltar que o controlador inteligente também pode ser desenvolvido em linguagens de programação de baixo nível, melhorando a eficiência de processamento do algoritmo Sistema de Formigas. Essa ideia é citada como possibilidade para trabalhos futuros, pois encontrar a melhor performance possível do controlador não é o principal objetivo deste trabalho.

3.3.4 PARÂMETROS DO CONTROLADOR

Essa seção expõe informações sobre testes realizados para selecionar os parâmetros do algoritmo Sistema de Formigas, posto que o desempenho e eficiência do algoritmo estão intimamente ligados aos parâmetros escolhidos, conforme visto na seção 2.7. Tais parâmetros

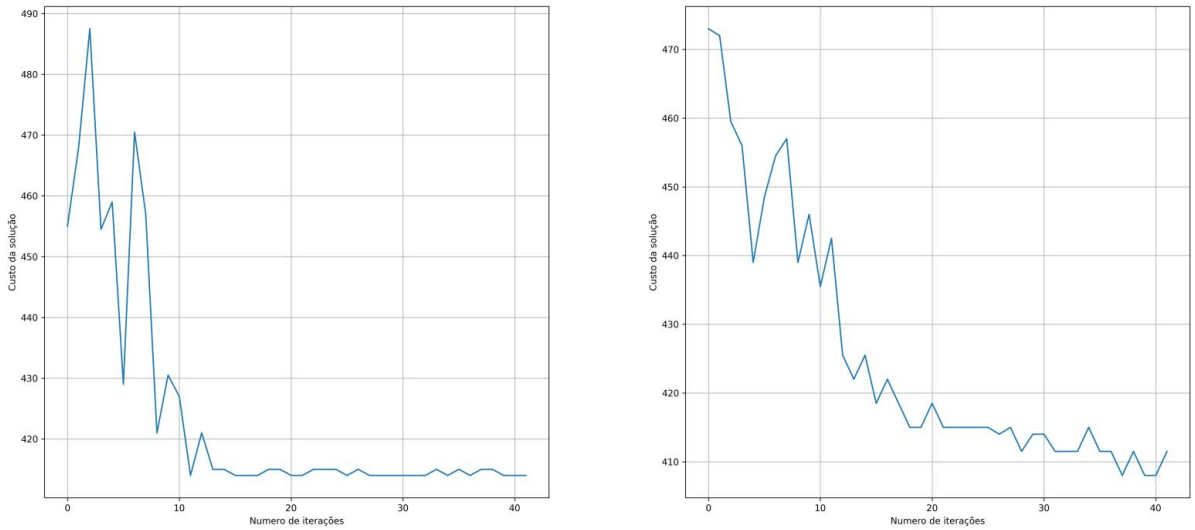
incluem informação heurística, influência dos feromônios, número de formigas e fator de evaporação. O método de seleção adotado para este trabalho, foi de tentativa e erro.

Para seleção dos parâmetros, foram realizadas diversas simulações aplicando uma demanda de 7 engrenagens A e 5 engrenagens B. Em cada simulação os valores de α e β foram variados com incrementos de 0.01 em α e decremento de 0.01 em β. α encontra-se iniciando em 0.01 e β em 0.99, mantendo constante em 0.37 o fator de evaporação ρ, e o feromônio Q excretado por cada formiga em 1. Então, em cada experimento das combinações de α e β, o número de formigas k foi variado com intervalo 1 a 70, com condição de parada em 40 iterações do algoritmo ou 50 segundos de simulação.

Consequentemente, cada simulação gerou um gráfico do custo em função do número de iterações do algoritmo, logo, os gráficos foram analisados de forma minuciosa, investigando quais obtiveram as melhores performances em relação aos custos, convergência para a resposta ótima em menos iterações e, mais importante, considerar soluções que foram flexíveis o suficiente para não ficar presas em ótimos locais.

Os gráficos das figuras 18, 19 e 20 mostram os principais comportamentos que foram observados durante a análise gráfica para decidir que os parâmetros escolhidos resultaram em uma boa performance do algoritmo.

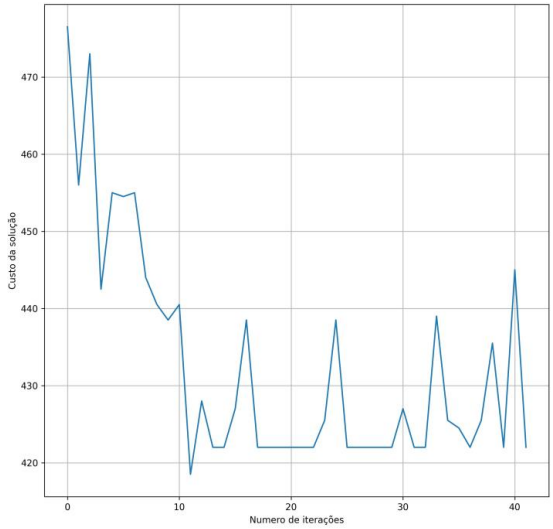
A Figura 18a, mostra um comportamento de convergência rápido, enquanto a Figura 18b apresenta uma convergência mais lenta, obtendo menores custos. Já nas figuras 19a e 19b, podem ser visualizadas situações em que as soluções acabaram ficando presas em ótimos locais. Por fim, as figuras 20a e 20b apresentam um comportamento bastante diversificado e flexível.



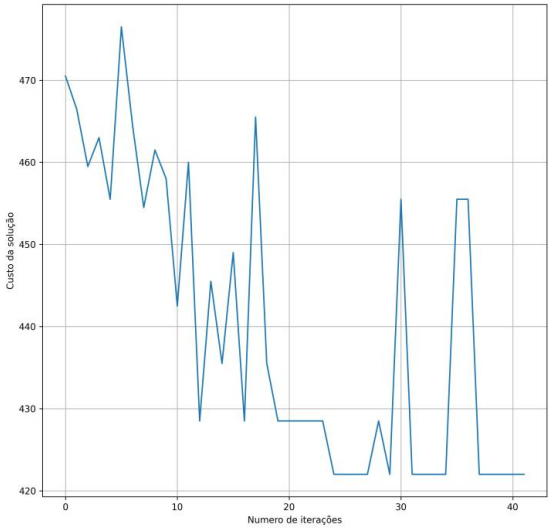
(a) Solução convergindo rapidamente **(b) Solução convergindo mais lentamente**

Figura 18 – Comportamento de convergência

Fonte: Autoria própria

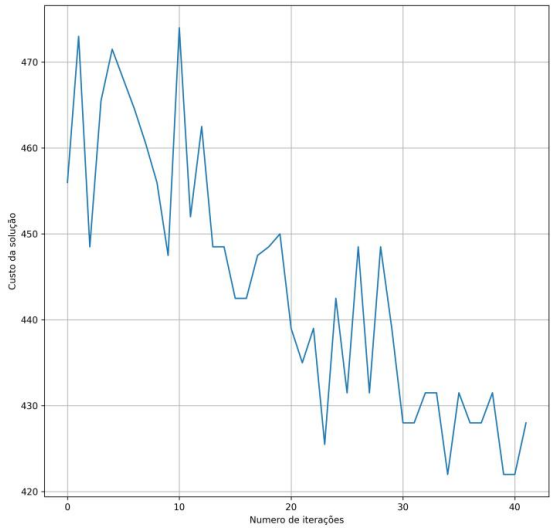


(a) Solução presa em ótimos locais com convergência rápida

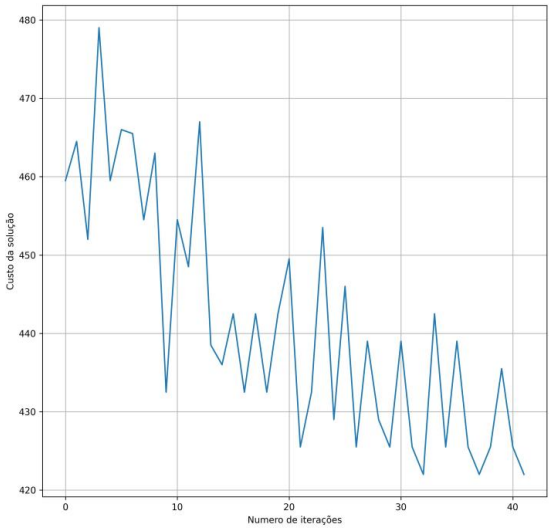


(b) Solução presa em ótimos locais com convergência mais lenta

Figura 19 – Comportamento de solução presa em ótimos locais
Fonte: Autoria própria



(a) Solução com bom nível de diversificação e flexibilidade



(b) Solução com bom nível de diversificação e flexibilidade

Figura 20 – Comportamento com diversificação e flexibilidade
Fonte: Autoria própria

Uma vez realizado a análise gráfica, foi possível identificar que os parâmetros fixados dentro de uma determinada faixa de valor, acabavam por gerar soluções aproximadamente similares, sendo que os parâmetros: $\alpha : 0.42$, $\beta : 0.58$, $k : 43$, $Q : 1$, $\rho : 0.37$, foram escolhidos dentre os parâmetros que obtiveram as melhores performances em relação a flexibilidade para escapar de ótimos locais, convergência rápida e custo final da solução.

É importante lembrar que o parâmetros escolhidos estabelecem uma boa performance ao controlador inteligente somente para o exemplo prático explorado. Se, eventualmente, tal

controlador for implementando em outro cenário de produção, testes para obter novos parâmetros devem ser realizados.

3.3.5 ILUSTRAÇÃO DO FUNCIONAMENTO DO CONTROLADOR INTELIGENTE

Uma vez realizadas as etapas de obtenção do sistema de controle, ajuste do espaço de estado do supervisor, implementação e teste dos parâmetros do Sistema de Formigas, é possível implementar e obter um controlador inteligente totalmente funcional. Para ilustração do funcionamento do controlador foi realizada uma sequência de duas simulações, contendo como parâmetros de entrada uma pequena demanda de engrenagens A e B.

Após o controlador inteligente receber essas demandas, o planejamento da produção é calculado e retornado por meio de um conjunto de eventos. Tais eventos, se habilitados sequencialmente no *hardware* dos agentes, são capazes de conduzir com segurança as atividades de manufatura para a demanda de produtos imposta.

Mais detalhes do funcionamento do controlador podem ser encontrados no vídeo (CONTROLADOR. . . , 2022), que fornece uma breve explicação da implementação e, em sequência, a simulação de dois testes. Já para verificação do código implementado, o leitor deve se dirigir ao *github* (SISTEMA-DE-FORMIGAS. . . , 2022).

3.4 SIMULAÇÃO E COLETA DE DADOS

Para simulação e coleta de dados foram passados alguns testes ao controlador inteligente. Neste teste, o controlador encontrou as melhores sequências de eventos para três tipos distintos de demanda de produção: alta, média e baixa. A demanda baixa refere-se à produção de 44 produtos A e 36 produtos B, enquanto as demandas média e alta são respectivamente caracterizadas por serem três e seis vezes maiores que o volume da demanda baixa.

Os parâmetros utilizados no controlador global foram os obtidos na análise gráfica na seção 3.3.4, exceto o número de iterações, os quais foram definidos em 50. Então, foram executadas 100 simulações computacionais para cada situação de produção. A Figura 21 ilustra o custo obtido em cada iteração do algoritmo, para as três melhores soluções encontradas em cada demanda.

Os dados coletados das simulações sugerem ser factível reduzir os custos de produção por meio da utilização do controlador inteligente. Neste caso, como o número de iterações

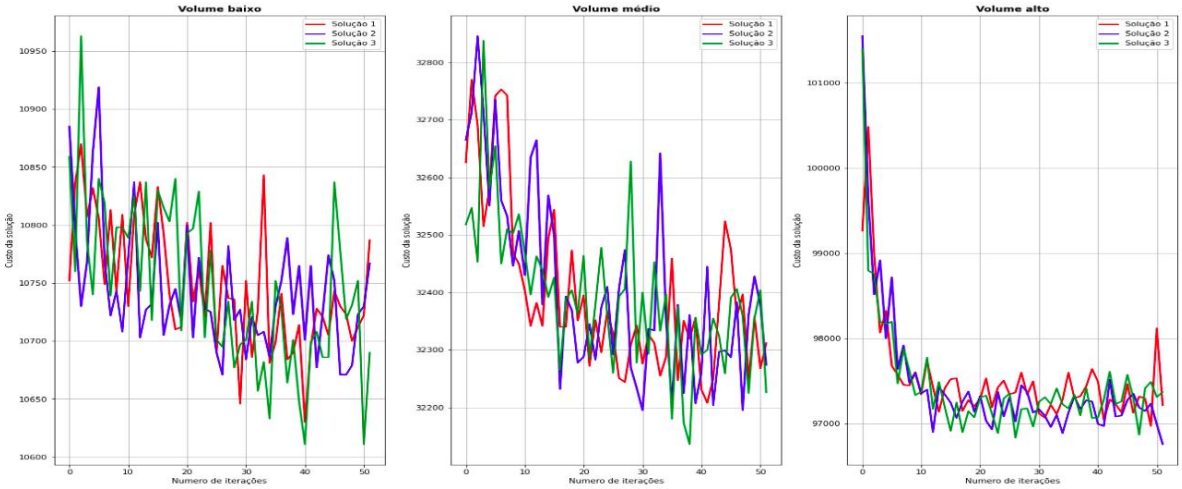


Figura 21 – Custo da solução x Iteração da Meta-heurística
Fonte: Autoria própria

definidas nas simulações foram relativamente baixas, observa-se que as soluções podem ser melhoradas.

Para melhorar o desempenho do controlador em relação aos custos, algumas abordagens podem ser utilizadas. Por exemplo, pode-se optar por aumentar o número de iterações do algoritmo, uma vez que observa-se que as soluções estavam convergindo para custos cada vez menores e, ainda não tinham alcançado um estado de estagnação. Porém, em contra partida, seria necessário implementar o controlador em um *hardware* com maior poder computacional, para garantir boas soluções em tempo viável.

Outro fato observado é que as menores demandas realmente atingem menores custos de produção, o que é trivial, pois o algoritmo percorre menos vezes o espaço de estados, diminuindo a complexidade computacional do problema de busca. Ou seja, caso o controlador seja implementado em um *hardware* com pouco poder computacional, pode ser interessante dividir a demanda em partes menores, para então, obter uma solução de baixo custo e utilizá-la repetidas vezes até alcançar a demanda especificada.

As melhores soluções encontradas para as três demandas geram um comportamento que impõe uma determinada quantidade de engrenagens a ser manufaturada por cada agente. A Tabela 2 traz esse comportamento, sendo possível observar sinais da ocorrência de cooperação entre os agentes, pois o controlador inteligente escalonou as atividades dividindo em aproximadamente 50% a quantidade de engrenagens feitas por agente.

Situação	Produto	Demanda	Qtd. feita pelo agente 1	Qtd. feita pelo agente 2
Alta	A	396	195	201
	B	321	161	160
Média	A	132	68	64
	B	107	52	55
Baixa	A	44	20	24
	B	36	20	16

Tabela 2 – Comportamento de manufatura, devido a melhor solução encontrada para cada demanda
Fonte: Autoria própria

Outra forma de tentar compreender se o controlador global está gerando cooperação entre os agentes é analisar a dispersão da quantidade de engrenagens que os agentes devem manufaturar em cada solução encontrada. Os gráficos da Figura 22, trazem dados de uma amostra de 100 simulações para demanda baixa. No eixo x, observa-se a quantidade de engrenagens que o agente precisou manufaturar e, no eixo y, a frequência de ocorrências dessa quantidade.

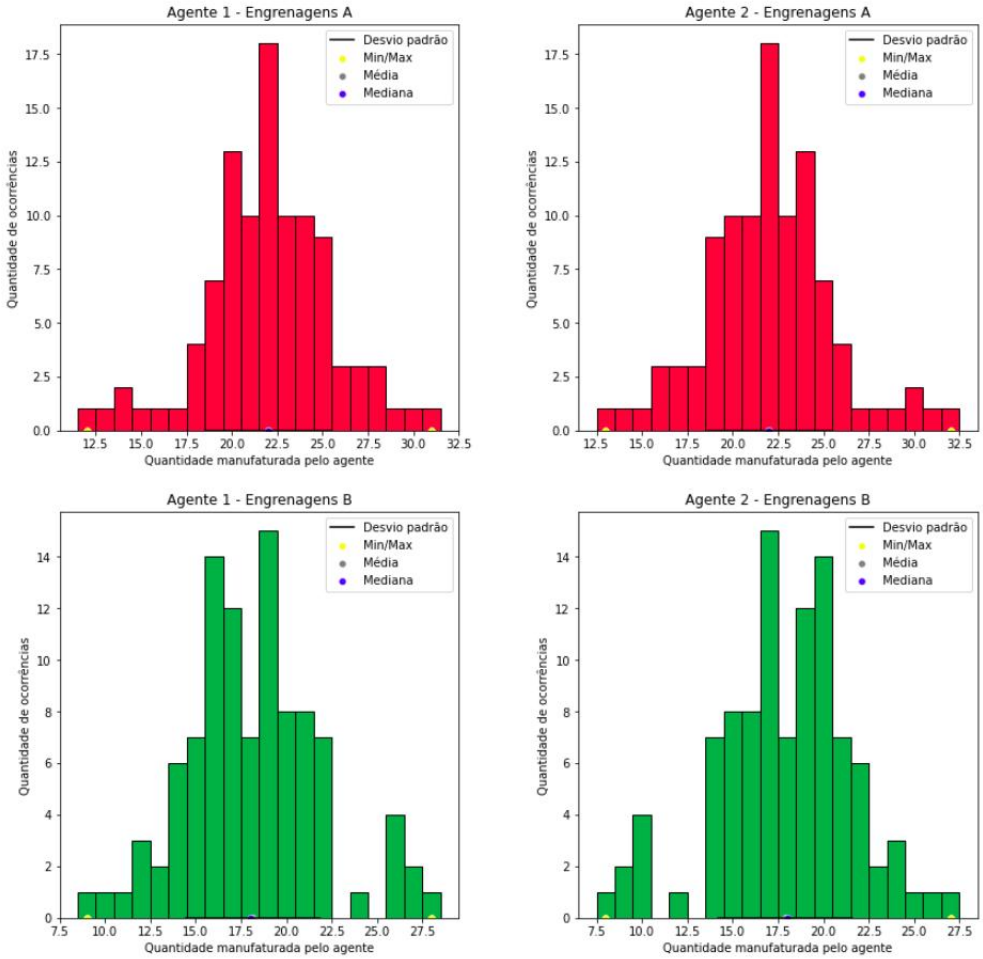


Figura 22 – Dispersão da quantidade de produtos feita por agente
Fonte: Autoria própria

É possível observar nos gráficos da Figura 22 que a distribuição da quantidade de engrenagens A e B a ser manufaturada, tem maiores casos de ocorrência, próximo de 22 e 18

unidades representando, respectivamente 50% da demanda de produção estimada. Para essa amostra foi calculado um desvio padrão de aproximadamente 4 unidades. Isto é, neste caso, a quantidade de engrenagens a ser manufaturada por agente tem um desvio de 4 unidades do que seria a divisão ideal de trabalho para cada agente produzir metade da demanda.

Os dados apresentados na Figura 22, reforçam a possibilidade do controlador global gerar cooperação entre os agentes, pois agrega potencial para distribuir a carga de trabalho. Pode-se notar ainda que, neste caso, o controlador tem uma leve inclinação por optar que o agente 2 manufature mais engrenagens A e o agente 1 engrenagens B. Esse fenômeno pode ter sido gerado pela escolha dos parâmetros configurados no Sistema de Formigas, ou por causa das restrições que o cenário impõe, abrindo espaço para futuras pesquisas complementares.

Para investigar um pouco mais a fundo se o controlador inteligente realmente tem potencial de instituir a cooperação de trabalho entre os agentes, foram realizados testes para uma amostra contendo 200 simulações. Dentro dessa amostra, foram passadas demandas aleatórias de engrenagens A e B, respeitando a seguinte regra: $46 \leq A \leq 71$ e $38 \leq B \leq 64$. Os gráficos da Figura 23, ilustram a distribuição das frequências de trabalho instituídas aos agentes.

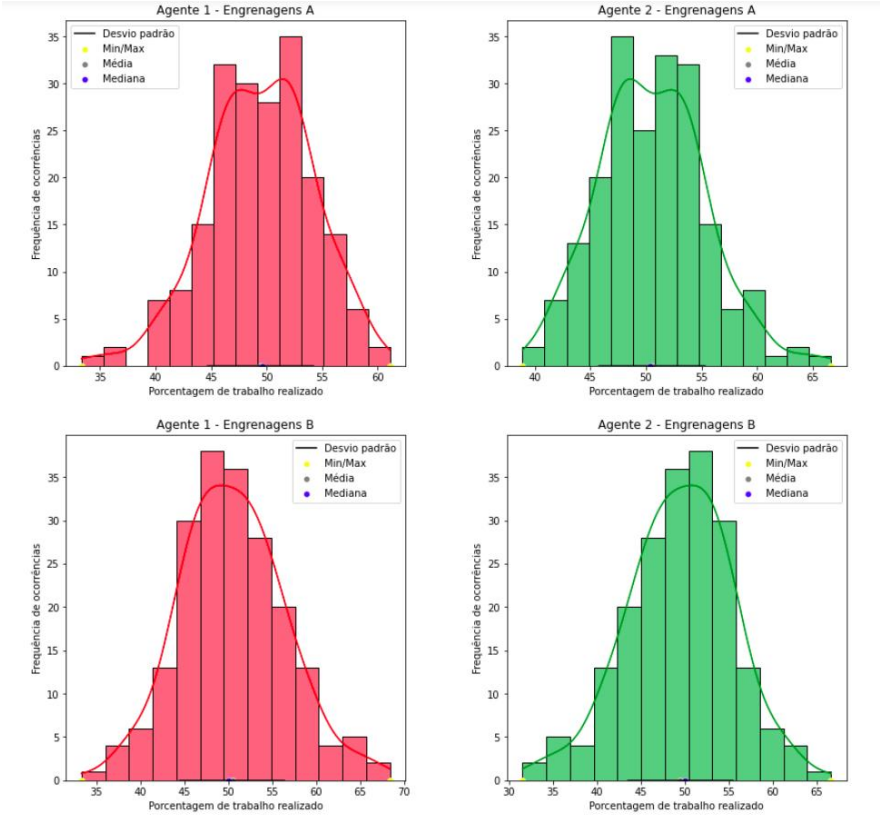


Figura 23 – Dispersão da porcentagem de trabalho realizado por agente
Fonte: Autoria própria

Nota-se que os gráficos se assemelham muito à uma distribuição simétrica, contendo a partir do centro a maioria das suas ocorrências. Observa-se também que a média de trabalho realizado por agente se aproxima de 50%, com um desvio padrão de aproximadamente 5%. Ou seja, grande parte das soluções encontradas pelo controlador inteligente divide, nessa amostra, a carga de trabalho em torno de 45% à 55% por agente, mostrando potencial de fazer os agentes cooperarem para cumprir uma determina demanda.

Outro ponto de interesse deste trabalho é fazer o controlador inteligente impor um comportamento flexível aos agentes e, para verificar esse aspecto, foram simuladas produções de duas pequenas demandas iguais, contendo respectivamente 5 engrenagens do tipo A e 7 engrenagens do tipo B.

Durante as duas simulações o controlador inteligente retornou soluções com sequências de eventos diferentes, que podem ser observadas na Tabela 5 Apêndice D. As figuras 24 e 25 expressam visualmente as áreas acessadas pelos agentes em função de cada evento habilitado pela sequência retornada pelo controlador em cada solução.

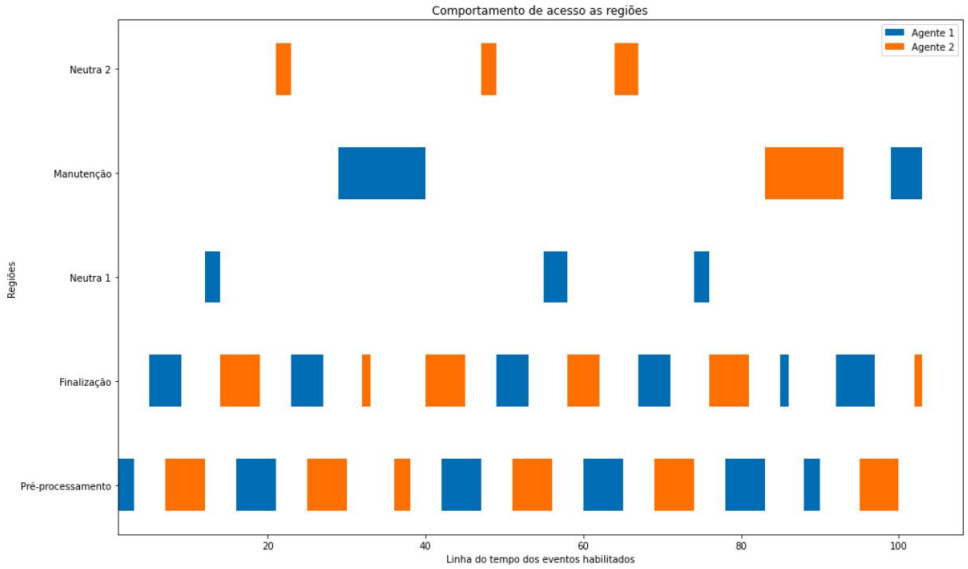


Figura 24 – Comportamento instituído aos agentes devido a simulação 1
Fonte: Autoria própria

Observa-se que as soluções geradas estabelecem comportamentos distintos, expondo o potencial de flexibilidade agregado pelo controlador inteligente, pois os eventos das soluções, quando habilitados, executam a produção de demandas iguais, porém, com diferentes rotinas de trabalho.

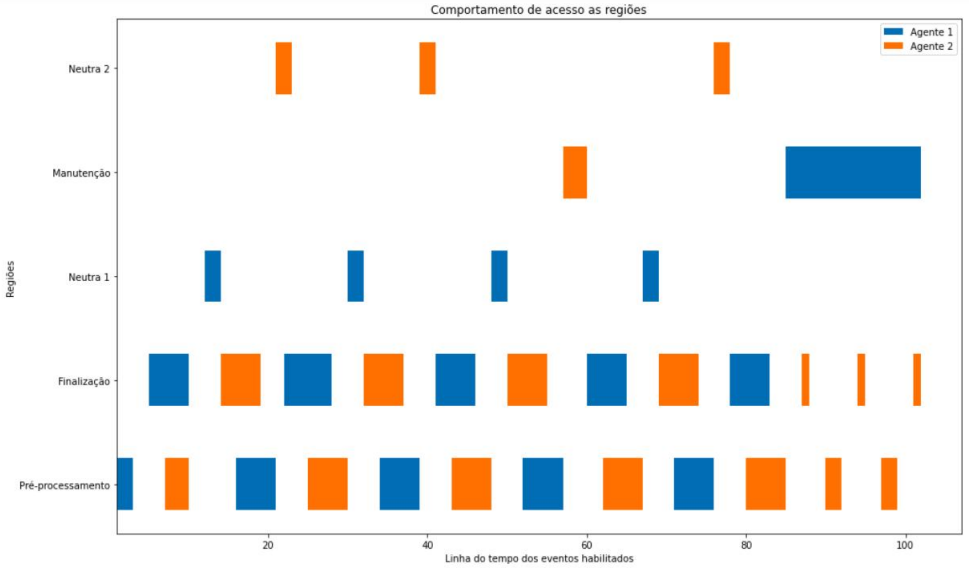


Figura 25 – Comportamento instituído aos agentes devido a simulação 2
Fonte: Autoria própria

3.5 TRABALHOS FUTUROS

O controlador inteligente construído pode ser utilizado para a implementação de outras aplicações industriais voltadas à otimização, ficando a cargo do pesquisador ajustar a função objetivo do controlador e integrar no sistema de controle. Dentre as pesquisas conjecturadas para a sequência, inclui-se:

- Criação de uma interface gráfica para o gerenciamento de produção e monitoramento do controlador global.
- Melhorias na função objetivo, inserindo mais informações, por exemplo, considerar tempo aproximado de execução dos processos.
- Introdução de escolhas dinâmicas, ou seja, após gerar uma sequência de produção, ser possível alterar as ações em tempo real, de acordo com o atual contexto dos agentes.
- Aplicação do controlador em um sistema físico.
- Criação de uma metodologia para ditar quais eventos podem ser abstraídos.
- Por fim, utilização de outras abordagens mais eficientes para a síntese do supervisor via TCS para a implementação do controlador.

4 CONCLUSÕES

Os resultados obtidos neste trabalho reforçam o potencial da aplicação de algoritmos meta-heurísticos em conjunto da teoria de controle supervisório para construção de aplicações de produção industrial, pois agregam características que garantem segurança à produção, seja a nível lógico ou de infraestrutura de *software* e *hardware*. Além disso, abrem espaço para sistemas de controle otimizáveis e flexíveis, aspectos que são, por sua vez, extremamente visados em ambientes modernos de produção denominados Indústria 4.0.

Em uma implementação de controle convencional, o que se deseja é encontrar uma boa sequência de eventos que retratam o funcionamento desejado do sistema, com essa sequência sendo, então, implementada em um CLP. Porém, neste modo de implementação não há um escalonamento de eventos e, consequentemente, as ações realizadas sob controle do CLP são meramente repetitivas e desprovidas de qualquer cooperação por parte dos dispositivos.

Os ganhos obtidos com a implementação do controlador inteligente, no que se refere à abordagem convencional de controle, são enormes, tendo em vista que o sistema de controle convencional implementa uma sequência que coordena os dispositivos repetindo-se infinitamente, sem considerar possíveis mudanças de contexto.

A integração do Sistema de Formigas ao controlador inteligente proposto neste trabalho trouxe uma série de benefícios ao caso prático abordado, dentre os quais se encontram: a possibilidade da redução de custos; a disponibilidade da alteração do contexto para manufaturar itens customizados; a flexibilização da produção adaptando o sistema de acordo com as demandas; e a criação de cooperação entre os agentes em chão de fábrica.

Outro aspecto importante vem da abstração utilizada na modelagem, que fornece a vantagem de um supervisor mais compacto. Em contrapartida, aspectos como abstração, controle local e observabilidade parcial requerem maior esforço para a construção do sistema, podendo se tornar um inconveniente. Dessa forma, visivelmente, a abstração reduz a complexidade de processamento do Sistema de Formigas e do seu espaço de busca, ficando como perspectiva para futuras pesquisas quantificar quão significativos podem ser os ganhos com a utilização da abstração em controladores.

Vale salientar que o controlador global pode ser construído com um supervisor sem abstração. Como ponto negativo, tem-se o aumento do espaço de estados e, consequentemente, o aumento da complexidade computacional para encontrar boas soluções.

O sistema de controle foi obtido por meio do cálculo de um supervisor monolítico.

Porém, o controlador inteligente pode ser implementado utilizando a abordagem monolítica para o cálculo do supervisor ou outras abordagens advindas da TCS. Por fim, espera-se que este trabalho sirva de suporte para que outras pessoas possam utilizar a abordagem e modelo expostos, para fins de desenvolvimento de novas soluções de sistemas de controle inteligentes, que são tao requisitados pela industria.

REFERÊNCIAS

AFZALIAN, A.; NOORBAKHSH, M.; NABAVI, S. PLC implementation of decentralized supervisory control for dynamic flow controller. In: IEEE. 2008 IEEE International Conference on Control Applications. [*S. l.: s. n.*], 2008. P. 522–527.

AKESSON, K. *et al.* **Supremica**. [*S. l.*], 2019. Disponível em: http://www.supremica.org/.

ALMEDER, C.; MÖNCH, L. Metaheuristics for scheduling jobs with incompatible families on parallel batching machines. **Journal of the Operational Research Society**, Springer, v. 62, n. 12, p. 2083–2096, 2011.

ALSZER, S.; KRYSTEK, J. Modular, didactic flexible manufacturing system: Case study. In: 2018 4th Int. Conf. on Control, Automation and Robotics (ICCAR). [*S. l.: s. n.*], abr. 2018. P. 121–125.

ALVES, L. V.; PENA, P. N.; TAKAHASHI, R. H. Planning on Discrete Event Systems using parallelism maximization. **Control Engineering Practice**, Elsevier, v. 112, p. 104813, 2021.

BANGEMANN, T. *et al.* Integration of Classical Components Into Industrial Cyber-Physical Systems. **Proceedings of the IEEE**, v. 104, n. 5, p. 947–959, mai. 2016.

BOLCH, G. *et al.* **Queueing Networks and Markov Chains. Modeling and Performance Evaluation with Computer Science Applications**. 2. ed. New Jersey: John Wiley & Sons, Inc., 2006.

CANCIO, G. A. D. **Uma arquitetura para a implementação de controle supervisório comunicante**. 2022. F. 50. Monografia (Trabalho de Conclusão de Curso) – Graduação em Engenharia de Computação, Universidade Tecnológica Federal do Paraná, Pato Branco.

CASSANDRAS, C. G.; LAFORTUNE, S. **Introduction to discrete event systems**. [*S. l.*]: Springer Science & Business Media, 2009.

CONTROLADOR inteligente - Rafael Anderson Dalmolin. [*S. l.*], 2022. Disponível em: https://drive.google.com/file/d/1R54-v74_xmc-9R2Qiojt-KIfnMG75p9V/view?usp=sharing.

CURY, J. E. R. Teoria de Controle Supervisório de Sistemas a Eventos Discretos. **V Simpósio Brasileiro de Automação Inteligente**, 2001.

DORIGO, M.; GAMBARDELLA, L. Ant colony system: a cooperative learning approach to the traveling salesman problem. **IEEE Transactions on Evolutionary Computation**, v. 1, n. 1, p. 53–66, 1997. DOI: 10.1109/4235.585892.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 26, n. 1, p. 29–41, 1996. DOI: 10.1109/3477.484436.

DORIGO, M.; BIRATTARI, M. *et al.* **Ant Colony Optimization and Swarm Intelligence: 4th International Workshop, ANTS 2004**. [*S. l.: s. n.*], jan. 2004. DOI: 10.1007/b99492.

DORIGO, M.; DI CARO, G.; GAMBARDELLA, L. M. Ant algorithms for discrete optimization. **Artificial life**, MIT Press, v. 5, n. 2, p. 137–172, 1999.

DRATH, R.; HORCH, A. Industrie 4.0: Hit or Hype? [Industry Forum]. **IEEE Industrial Electronics Magazine**, v. 8, n. 2, p. 56–58, 2014.

ESMAEILIAN, B.; BEHDAD, S.; WANG, B. The evolution and future of manufacturing: A review. **Journal of Manufacturing Systems**, v. 39, p. 79–100, 2016.

FERROLHO, A.; CRISOSTOMO, M. Intelligent Control and Integration Software for Flexible Manufacturing Cells. **IEEE Transactions on Industrial Informatics**, v. 3, n. 1, p. 3–11, fev. 2007.

GAREY, M. R.; JOHNSON, D. S. **Computers and intractability**. [*S. l.*]: freeman San Francisco, 1979. v. 174.

GOLDBARG, E.; GOLDBARG, M.; LUNA, H. **Otimização combinatória e metaheurısticas: algoritmos e apliacações**. [*S. l.*]: Elsevier Brasil, 2017.

GROOVER, M. **Automação Industrial e Sistemas de Manufatura**. 3. ed. São Paulo: Pearson, 2011.

HACKLER, D.; SIME, D. G.; WALD, S. F. Enabling Electronics With Physically Flexible ICs and Hybrid Manufacturing. **Proceedings of the IEEE**, v. 103, n. 4, p. 633–643, abr. 2015.

HARRISON, R.; VERA, D.; AHMAD, B. Engineering Methods and Tools for Cyber-Physical Automation Systems. **Proceedings of the IEEE**, v. 104, n. 5, p. 973–985, mai. 2016.

HOPCROFT, J. E.; ULLMAN, J. D.; MOTWANI, R. **Introduction to Automata Theory, Languages and Computation**. 2. ed. [*S. l.*]: Addison Wesley, 2001.

HOPCROFT, J. E.; MOTWANI, R.; ULLMAN, J. D. **Introduction to Automata Theory, Languages, and Computation**. 2. ed. United States of America: Pearson Education, 1979.

JUPYTER Notebook. [*S. l.*], 2022. Disponível em: https://jupyter.org/.

LEAL, A. B.; DA CRUZ, D. L.; HOUNSELL, M. d. S. Supervisory control implementation into programmable logic controllers. In: IEEE. 2009 IEEE Conference on Emerging Technologies & Factory Automation. [*S. l.: s. n.*], 2009. P. 1–7.

LIU, Y. *et al.* Review on cyber-physical systems. **IEEE/CAA Journal of Automatica Sinica**, v. 4, n. 1, p. 27–40, jan. 2017.

MARCUS Aurelius. [*S. l.*].

MURATA, T. Petri Nets: Properties, Analysis and Applications. **Proc. of the IEEE, v.77**, p. 541–580, 1989.

OGATA, K. **Discrete-time Control Systems**. 2. ed. University of Minnesota: Prentice-Hall International, Inc., 1996.

PENA, P. N. *et al.* Control of flexible manufacturing systems under model uncertainty using supervisory control theory and evolutionary computation schedule synthesis. **Information Sciences**, Elsevier, v. 329, p. 491–502, 2016.

PYTHON. [*S. l.*], 2022. Disponível em: https://www.python.org/.

RAMADGE, P. J.; WONHAM, W. M. The control of discrete event systems. **Proceedings of the IEEE**, IEEE, v. 77, n. 1, p. 81–98, 1989.

REDDY, B. M. Y. J. **Industrial Process Automation Systems**. 1. ed. United kingdom: Elsevier Inc, 2014.

SALDIVAR, A. A. F. *et al.* Industry 4.0 with cyber-physical integration: A design and manufacture perspective. In: 2015 21st International Conference on Automation and Computing (ICAC). [*S. l.: s. n.*], set. 2015. P. 1–6.

SANTOS, A.; DOURADO, A. Global optimization of energy and production in process industries: a genetic algorithm application. **Control engineering practice**, Elsevier, v. 7, n. 4, p. 549–554, 1999.

SILVA, A. L.; RIBEIRO, R.; TEIXEIRA, M. Modeling and control of flexible context-dependent manufacturing systems. **Information Sciences**, v. 421, p. 1–14, 2017.

SIPSER, M. **Introdução à Teoria da Computação**. 2. ed. [*S. l.*]: Cengage Learning, 2012.

SISTEMA-DE-FORMIGAS. [*S. l.*], 2022. Disponível em: https://github.com/RafaelAndersonDalmolin/CONTROLE-SUPERVISORIO-COOPERATIVO-EMPREGANDO-A-META-HEURISTICA-SISTEMA-DE-FORMIGAS.

SONG, Y. *et al.* Bottleneck station scheduling in semiconductor assembly and test manufacturing using ant colony optimization. **IEEE Transactions on Automation Science and Engineering**, IEEE, v. 4, n. 4, p. 569–578, 2007.

TALBI, E.-G. **Metaheuristics: from design to implementation**. [*S. l.*]: John Wiley & Sons, 2009. v. 74.

TEIXEIRA, M. **Explorando o uso de distinguidores e de autômatos finitos estendidos na teoria do controle supervisório de sistemas a eventos discretos**. 2013. Tese (Doutorado) – Universidade Federal de Santa Catarina.

TRAPPEY, A. J. C. *et al.* A Review of Technology Standards and Patent Portfolios for Enabling Cyber-Physical Systems in Advanced Manufacturing. **IEEE Access**, v. 4, p. 7356–7382, 2016.

WANG, W.; YUAN, C.; LIU, X. A fuzzy approach to multi-product mixed production job shop scheduling algorithm. In: IEEE. 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery. [*S. l.: s. n.*], 2008. v. 1, p. 95–99.

WEN, S.; GUO, G. Control and resource allocation of cyber-physical systems. **IET Control Theory Applications**, v. 10, n. 16, p. 2038–2048, 2016.

WONHAM, W. **Notes on Discrete Event Systems**. [*S. l.: s. n.*], 2002. University of Toronto.

YOUSIF, M. Manufacturing and the Cloud. **IEEE Cloud Computing**, v. 3, n. 4, p. 4–5, jul. 2016.

ZIELINSKI, K. M. *et al.* Flexible control of Discrete Event Systems using environment simulation and Reinforcement Learning. **Applied Soft Computing**, Elsevier, v. 111, p. 107714, 2021.

APÊNDICE A — DISPONIBILIDADE DO CÓDIGO

Para a verificação do código que implementa o controlador inteligente, além das simulações realizadas, o leitor deve se direcionar ao repositório (SISTEMA-DE-FORMIGAS. . . , 2022). Além do código que pode ser clonado para estudo ou até mesmo adição de novas funcionalidades, este repositório possui uma ampla documentação com todas as etapas para utilizar o controlador.

APÊNDICE B — EVENTOS DO AUTÔMATO G_1

Rótulo do evento	Controlabilidade	Custo	Descrição
B1_FIN_A	Sim	1.0	Sai da região de preparação em direção a área de finalização
B1_FIN_A_END	Não	45.0	Chegou na área de finalização com o produto do tipo A
B1_FIN_B	Sim	1.0	Sai da região de preparação em direção a área de finalização
B1_FIN_B_END	Não	30.0	Chegou na área de finalização com o produto do tipo B
B1_MAINTENANCE	Não	20.0	Abstrai eventos que realizam a manutenção
B1_POINT_OF_INTEREST_FIN_A	Não	50.0	Abstrai eventos que finalizam o produto do tipo A
B1_POINT_OF_INTEREST_FIN_B	Não	35.0	Abstrai eventos que finalizam o produto do tipo B
B1_POINT_OF_INTEREST_PRE_A	Não	30.0	Abstrai eventos que preparam o produto do tipo A
B1_POINT_OF_INTEREST_PRE_B	Não	15.0	Abstrai eventos que preparam o produto do tipo B
B1_PRE	Sim	1.0	Sai da região neutra em direção a área de preparação
B1_PREPARATION_A	Sim	1.0	Inicia o processo de preparação do produto A
B1_PREPARATION_B	Sim	1.0	Inicia o processo de preparação do produto B
B1_PRE_A	Sim	1.0	Após a finalização de um produto A, sai em direção a área de preparação
B1_PRE_B	Sim	1.0	Após a finalização de um produto B, sai em direção a área de preparação
B1_PRE_END	Não	10.0	Chegou na área de preparação
B1_STOP	Sim	1.0	Sai da região de finalização em direção a região neutra
B1_STOP_END	Não	8.0	Chegou na região neutra
B1_SUPPORT	Sim	1.0	Sai da região de finalização em direção a manutenção
B1_SUPPORT_END	Não	10.0	Chegou na região de manutenção

Tabela 3 – Eventos do autômato G_1.
Fonte: Autoria própria

APÊNDICE C — EVENTOS DO AUTÔMATO G_2

Rótulo do evento	Controlabilidade	Custo	Descrição
B2_FIN_A	Sim	1.0	Sai da região de preparação em direção a área de finalização
B2_FIN_A_END	Não	45.0	Chegou na área de finalização com o produto do tipo A
B2_FIN_B	Sim	1.0	Sai da região de preparação em direção a área de finalização
B2_FIN_B_END	Não	30.0	Chegou na área de finalização com o produto do tipo B
B2_MAINTENANCE	Não	20.0	Abstrai eventos que realizam a manutenção
B2_POINT_OF_INTEREST_FIN_A	Não	50.0	Abstrai eventos que finalizam o produto do tipo A
B2_POINT_OF_INTEREST_FIN_B	Não	35.0	Abstrai eventos que finalizam o produto do tipo B
B2_POINT_OF_INTEREST_PRE_A	Não	30.0	Abstrai eventos que preparam o produto do tipo A
B2_POINT_OF_INTEREST_PRE_B	Não	15.0	Abstrai eventos que preparam o produto do tipo B
B2_PRE	Sim	1.0	Sai da região neutra em direção a área de preparação
B2_PREPARATION_A	Sim	1.0	Inicia o processo de preparação do produto A
B2_PREPARATION_B	Sim	1.0	Inicia o processo de preparação do produto B
B2_PRE_A	Sim	1.0	Após a finalização de um produto A, sai em direção a área de preparação
B2_PRE_B	Sim	1.0	Após a finalização de um produto B, sai em direção a área de preparação
B2_PRE_END	Não	10.0	Chegou na área de preparação
B2_STOP	Sim	1.0	Sai da região de finalização em direção a região neutra
B2_STOP_END	Não	8.0	Chegou na região neutra
B2_SUPPORT	Sim	1.0	Sai da região de finalização em direção a manutenção
B2_SUPPORT_END	Não	10.0	Chegou na região de manutenção

Tabela 4 – Eventos do autômato G_2.
Fonte: Autoria própria

APÊNDICE D — SEQUÊNCIAS DE EVENTOS GERADAS NAS SIMULAÇÕES

Id	Sequência de eventos gerada pela simulação 1	Sequência de eventos gerada pela simulação 2
0	B1_PRE	B1_PRE
1	B1_PRE_END	B1_PRE_END
2	B1_PREPARATION_A	B1_PREPARATION_B
3	B1_POINT_OF_INTEREST_PRE_A	B1_POINT_OF_INTEREST_PRE_B
4	B1_FIN_A	B1_FIN_B
5	B1_FIN_A_END	B1_FIN_B_END
6	B2_PRE	B2_PRE
7	B2_PRE_END	B2_PRE_END
8	B2_PREPARATION_B	B2_PREPARATION_A
9	B1_POINT_OF_INTEREST_FIN_A	B2_POINT_OF_INTEREST_PRE_A
10	B1_STOP	B1_POINT_OF_INTEREST_FIN_B
11	B2_POINT_OF_INTEREST_PRE_B	B1_STOP
12	B1_STOP_END	B1_STOP_END
13	B2_FIN_B	B2_FIN_A
14	B2_FIN_B_END	B2_FIN_A_END
15	B1_PRE	B1_PRE
16	B1_PRE_END	B1_PRE_END
17	B1_PREPARATION_B	B1_PREPARATION_B
18	B1_POINT_OF_INTEREST_PRE_B	B1_POINT_OF_INTEREST_PRE_B
19	B2_POINT_OF_INTEREST_FIN_B	B2_POINT_OF_INTEREST_FIN_A
20	B2_STOP	B2_STOP
21	B2_STOP_END	B2_STOP_END
22	B1_FIN_B	B1_FIN_B
23	B1_FIN_B_END	B1_FIN_B_END
24	B2_PRE	B2_PRE
25	B2_PRE_END	B2_PRE_END
26	B2_PREPARATION_A	B2_PREPARATION_A
27	B1_POINT_OF_INTEREST_FIN_B	B2_POINT_OF_INTEREST_PRE_A
28	B1_SUPPORT	B1_POINT_OF_INTEREST_FIN_B
29	B1_SUPPORT_END	B1_STOP
30	B2_POINT_OF_INTEREST_PRE_A	B1_STOP_END
31	B2_FIN_A	B2_FIN_A
32	B2_FIN_A_END	B2_FIN_A_END
33	B2_POINT_OF_INTEREST_FIN_A	B1_PRE
34	B2_PRE_A	B1_PRE_END
35	B1_MAINTENANCE	B1_PREPARATION_B

Id	Sequência de eventos gerada pela simulação 1	Sequência de eventos gerada pela simulação 2
36	B2_PRE_END	B1_POINT_OF_INTEREST_PRE_B
37	B2_PREPARATION_B	B2_POINT_OF_INTEREST_FIN_A
38	B2_POINT_OF_INTEREST_PRE_B	B2_STOP
39	B2_FIN_B	B2_STOP_END
40	B2_FIN_B_END	B1_FIN_B
41	B1_PRE	B1_FIN_B_END
42	B1_PRE_END	B2_PRE
43	B1_PREPARATION_B	B2_PRE_END
44	B1_POINT_OF_INTEREST_PRE_B	B2_PREPARATION_A
45	B2_POINT_OF_INTEREST_FIN_B	B2_POINT_OF_INTEREST_PRE_A
46	B2_STOP	B1_POINT_OF_INTEREST_FIN_B
47	B2_STOP_END	B1_STOP
48	B1_FIN_B	B1_STOP_END
49	B1_FIN_B_END	B2_FIN_A
50	B2_PRE	B2_FIN_A_END
51	B2_PRE_END	B1_PRE
52	B2_PREPARATION_A	B1_PRE_END
53	B1_POINT_OF_INTEREST_FIN_B	B1_PREPARATION_B
54	B1_STOP	B1_POINT_OF_INTEREST_PRE_B
55	B1_STOP_END	B2_POINT_OF_INTEREST_FIN_A
56	B2_POINT_OF_INTEREST_PRE_A	B2_SUPPORT
57	B2_FIN_A	B2_SUPPORT_END
58	B2_FIN_A_END	B1_FIN_B
59	B1_PRE	B2_MAINTENANCE
60	B1_PRE_END	B1_FIN_B_END
61	B1_PREPARATION_A	B2_PRE
62	B2_POINT_OF_INTEREST_FIN_A	B2_PRE_END
63	B2_STOP	B2_PREPARATION_A
64	B2_STOP_END	B2_POINT_OF_INTEREST_PRE_A
65	B1_POINT_OF_INTEREST_PRE_A	B1_POINT_OF_INTEREST_FIN_B
66	B1_FIN_A	B1_STOP

Id	Sequência de eventos gerada pela simulação 1	Sequência de eventos gerada pela simulação 2
67	B1_FIN_A_END	B1_STOP_END
68	B2_PRE	B2_FIN_A
69	B2_PRE_END	B2_FIN_A_END
70	B2_PREPARATION_B	B1_PRE
71	B1_POINT_OF_INTEREST_FIN_A	B1_PRE_END
72	B1_STOP	B1_PREPARATION_B
73	B2_POINT_OF_INTEREST_PRE_B	B1_POINT_OF_INTEREST_PRE_B
74	B1_STOP_END	B2_POINT_OF_INTEREST_FIN_A
75	B2_FIN_B	B2_STOP
76	B2_FIN_B_END	B2_STOP_END
77	B1_PRE	B1_FIN_B
78	B1_PRE_END	B1_FIN_B_END
79	B1_PREPARATION_B	B2_PRE
80	B1_POINT_OF_INTEREST_PRE_B	B2_PRE_END
81	B2_POINT_OF_INTEREST_FIN_B	B2_PREPARATION_A
82	B2_SUPPORT	B2_POINT_OF_INTEREST_PRE_A
83	B2_SUPPORT_END	B1_POINT_OF_INTEREST_FIN_B
84	B1_FIN_B	B1_SUPPORT
85	B1_FIN_B_END	B1_SUPPORT_END
86	B1_POINT_OF_INTEREST_FIN_B	B2_FIN_A
87	B1_PRE_B	B2_FIN_A_END
88	B1_PRE_END	B2_POINT_OF_INTEREST_FIN_A
89	B1_PREPARATION_B	B2_PRE_A
90	B1_POINT_OF_INTEREST_PRE_B	B2_PRE_END
91	B1_FIN_B	B2_PREPARATION_B
92	B1_FIN_B_END	B2_POINT_OF_INTEREST_PRE_B
93	B2_MAINTENANCE	B2_FIN_B
94	B2_PRE	B2_FIN_B_END
95	B2_PRE_END	B2_POINT_OF_INTEREST_FIN_B
96	B2_PREPARATION_A	B2_PRE_B
97	B1_POINT_OF_INTEREST_FIN_B	B2_PRE_END
98	B1_SUPPORT	B2_PREPARATION_B
99	B1_SUPPORT_END	B2_POINT_OF_INTEREST_PRE_B
100	B2_POINT_OF_INTEREST_PRE_A	B2_FIN_B
101	B2_FIN_A	B2_FIN_B_END
102	B2_FIN_A_END	B2_POINT_OF_INTEREST_FIN_B
103	B2_POINT_OF_INTEREST_FIN_A	

Tabela 5 – Sequências de eventos geradas nas simulações
Fonte: Autoria própria

ANEXO A — LEI N.º 9.610, DE 19 DE FEVEREIRO DE 1998: DIREITOS AUTORAIS / DISPOSIÇÕES PRELIMINARES

Presidência da República
Casa Civil
Subchefia para Assuntos Jurídicos

LEI Nº 9.610, DE 19 DE FEVEREIRO DE 1998.

Mensagem de veto

Regulamento

Altera, atualiza e consolida a legislação sobre direitos autorais e dá outras providências.

O PRESIDENTE DA REPÚBLICA Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

Título I

Disposições Preliminares

Art. 1º Esta Lei regula os direitos autorais, entendendo-se sob esta denominação os direitos de autor e os que lhes são conexos.

Art. 2º Os estrangeiros domiciliados no exterior gozarão da proteção assegurada nos acordos, convenções e tratados em vigor no Brasil.

Parágrafo único. Aplica-se o disposto nesta Lei aos nacionais ou pessoas domiciliadas em país que assegure aos brasileiros ou pessoas domiciliadas no Brasil a reciprocidade na proteção aos direitos autorais ou equivalentes.

Art. 3º Os direitos autorais reputam-se, para os efeitos legais, bens móveis.

Art. 4º Interpretam-se restritivamente os negócios jurídicos sobre os direitos autorais.

Art. 5º Para os efeitos desta Lei, considera-se:

I - publicação - o oferecimento de obra literária, artística ou científica ao conhecimento do público, com o consentimento do autor, ou de qualquer outro titular de direito de autor, por qualquer forma ou processo;

II - transmissão ou emissão - a difusão de sons ou de sons e imagens, por meio de ondas radioelétricas; sinais de satélite; fio, cabo ou outro condutor; meios óticos ou qualquer outro processo eletromagnético;

III - retransmissão - a emissão simultânea da transmissão de uma empresa por outra;

IV - distribuição - a colocação à disposição do público do original ou cópia de obras literárias, artísticas ou científicas, interpretações ou execuções fixadas e fonogramas, mediante a venda, locação ou qualquer outra forma de transferência de propriedade ou posse;

V - comunicação ao público - ato mediante o qual a obra é colocada ao alcance do público, por qualquer meio ou procedimento e que não consista na distribuição de exemplares;

VI - reprodução - a cópia de um ou vários exemplares de uma obra literária, artística ou científica ou de um fonograma, de qualquer forma tangível, incluindo qualquer armazenamento permanente ou temporário por meios eletrônicos ou qualquer outro meio de fixação que venha a ser desenvolvido;

VII - contrafação - a reprodução não autorizada;

VIII - obra:

a) em co-autoria - quando é criada em comum, por dois ou mais autores;

b) anônima - quando não se indica o nome do autor, por sua vontade ou por ser desconhecido;

c) pseudônima - quando o autor se oculta sob nome suposto;

d) inédita - a que não haja sido objeto de publicação;

e) póstuma - a que se publique após a morte do autor;

f) originária - a criação primígena;

g) derivada - a que, constituindo criação intelectual nova, resulta da transformação de obra originária;

h) coletiva - a criada por iniciativa, organização e responsabilidade de uma pessoa física ou jurídica, que a publica sob seu nome ou marca e que é constituída pela participação de diferentes autores, cujas contribuições se fundem numa criação autônoma;

i) audiovisual - a que resulta da fixação de imagens com ou sem som, que tenha a finalidade de criar, por meio de sua reprodução, a impressão de movimento, independentemente dos processos de sua captação, do suporte usado inicial ou posteriormente para fixá-lo, bem como dos meios utilizados para sua veiculação;

IX - fonograma - toda fixação de sons de uma execução ou interpretação ou de outros sons, ou de uma representação de sons que não seja uma fixação incluída em uma obra audiovisual;

X - editor - a pessoa física ou jurídica à qual se atribui o direito exclusivo de reprodução da obra e o dever de divulgá-la, nos limites previstos no contrato de edição;

XI - produtor - a pessoa física ou jurídica que toma a iniciativa e tem a responsabilidade econômica da primeira fixação do fonograma ou da obra audiovisual, qualquer que seja a natureza do suporte utilizado*;*

XII - radiodifusão - a transmissão sem fio, inclusive por satélites, de sons ou imagens e sons ou das representações desses, para recepção ao público e a transmissão de sinais codificados, quando os meios de decodificação sejam oferecidos ao público pelo organismo de radiodifusão ou com seu consentimento;

XIII - artistas intérpretes ou executantes - todos os atores, cantores, músicos, bailarinos ou outras pessoas que representem um papel, cantem, recitem, declamem, interpretem ou executem em qualquer forma obras literárias ou artísticas ou expressões do folclore.

XIV - titular originário - o autor de obra intelectual, o intérprete, o executante, o produtor fonográfico e as empresas de radiodifusão. (Incluído pela Lei nº 12.853, de 2013)

Art. 6º Não serão de domínio da União, dos Estados, do Distrito Federal ou dos Municípios as obras por eles simplesmente subvencionadas.

...

Texto completo da lei:

ÍNDICE REMISSIVO