Diagnóstico Síncrono Descentralizado com Coordenação ⋆

Patrícia C. Mayer * **Felipe G. Cabral** * **Marcos V. Moreira** **
João Pedro F. de Oliveira *

* *DAS - Departamento de Automação e Sistemas, Universidade Federal de Santa Catarina, SC, (e-mails: patricia.mayer@posgrad.ufsc.br, felipe.gomes.cabral@ufsc.br, joao.pedro.fragoso@grad.ufsc.br).*
** *COPPE - Programa de Engenharia Elétrica, Universidade Federal do Rio de Janeiro, RJ, (e-mail: moreira.mv@poli.ufrj.br).*

Abstract: Recently, a new architecture for decentralized diagnosis called Decentralized Synchronous Diagnosis has been proposed. In this scheme, local diagnosers are computed based on the fault-free behavior of the system components, which reduces the size of the diagnosers for implementation. Although this method has been successfully implemented, its main drawback is the growth of the fault-free language of the system for diagnosis, which reduces the diagnosis efficiency. In this work, in order to circumvent this problem, we propose a decentralized synchronous diagnosis with coordination (DSDC) that refines the diagnosis status using cluster automata of the local components. To do so, we also propose a communication protocol between local state estimators and the coordinator. We show that this method prevents the growth of the fault-free language for diagnosis, which guarantees the same diagnosis performance as the traditional centralized diagnosis method. In addition, to show the usability of the method, a practical implementation in a didactic manufacturing system is presented.

Resumo: Recentemente, uma nova arquitetura para diagnóstico descentralizado denominada Diagnóstico Síncrono Descentralizado foi proposta. Nesse método, diagnosticadores locais são calculados com base no comportamento livre de falha dos componentes do sistema, reduzindo o tamanho dos diagnosticadores para implementação. Embora esse método tenha sido implementado com sucesso, sua principal desvantagem é o crescimento da linguagem livre de falha gerada pelo sistema para o diagnóstico, o que reduz sua eficiência. Neste trabalho, com o intuito de contornar esse problema, um método para o diagnóstico síncrono descentralizado com coordenação (DSDC), que utiliza subautômatos em forma de *clusters* dos componentes locais para o refinamento do diagnóstico, é proposto. Para tanto, um protocolo de comunicação entre os estimadores de estado locais e o coordenador também é proposto. O método previne o crescimento da linguagem livre de falha para o diagnóstico, garantindo a mesma performance que o método tradicional centralizado. Além disso a aplicação do diagnóstico DSDC em uma planta didática de manufatura é apresentada com o objetivo de ilustrar sua implementação.

Keywords: Synchronous diagnosis; Decentralized diagnosis; Discrete Event Systems; Fault diagnosis; Automata.

Palavras-chaves: Diagnóstico síncrono; Diagnóstico descentralizado; Sistemas a Eventos Discretos; Diagnóstico de falhas; Autômatos.

1. INTRODUÇÃO

O diagnóstico de falhas é fundamental para a operação segura de sistemas de engenharia. Esse problema tem recebido considerável atenção na literatura no contexto de Sistemas a Eventos Discretos (SEDs) (Sampath et al., 1995; Debouk et al., 2000; Qiu e Kumar, 2006; Lefebvre e Delherm, 2007; Carvalho et al., 2012; Cabral et al., 2015; Cabral e Moreira, 2020). Em Sampath et al. (1995), um diagnosticador monolítico que recebe todas as informações dos sensores do sistema por meio de um único canal de comunicação é introduzido. Embora o diagnóstico obtido por esse método seja preciso, o cálculo desse diagnosticador é evitado porque, na análise de pior caso, o espaço de estados do diagnosticador cresce exponencialmente com a cardinalidade do espaço de estados do modelo da planta. Além disso, em diversas aplicações, a informação dos sensores está fisicamente distribuída e, nesses casos, outras arquiteturas para o diagnóstico, como a descentralizada (Debouk et al., 2000; Qiu e Kumar, 2006; Wang et al.,

⋆ O presente trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa e Inovação do Estado de Santa Catarina (FAPESC), da Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), do Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

2007) e a distribuída (Qiu e Kumar, 2005; Keroglou e Hadjicostis, 2018) são mais apropriadas.

Na arquitetura descentralizada proposta no Protocolo 3 de Debouk et al. (2000), diagnosticadores locais são calculados a partir do modelo da planta com base em observações locais de eventos. A falha é diagnosticada quando pelo menos um dos diagnosticadores indica sua ocorrência. Métodos polinomiais para a verificação da diagnosticabilidade descentralizada, chamada de codiagnosticabilidade, têm sido propostos na literatura (Moreira et al., 2011, 2016). Na arquitetura distribuída, os diagnosticadores locais podem se comunicar, trocando informações como a observação de eventos e estimativas de estado com o objetivo de refinar o diagnóstico (Keroglou e Hadjicostis, 2018). A principal desvantagem dessas arquiteturas é que os diagnosticadores locais são calculados com base no modelo global do sistema, cujo espaço de estados pode crescer exponencialmente com relação ao número de seus componentes (Cassandras e Lafortune, 2008), podendo gerar um alto custo computacional para o diagnóstico.

Com o intuito de evitar o uso do modelo global da planta para o diagnóstico de falhas, a arquitetura modular é proposta em Debouk et al. (2002); Contant et al. (2006). Nesses trabalhos, é suposto que o evento de falha está modelado em um único componente do sistema, e o objetivo é utilizar apenas esse componente para diagnosticar a falha. Nesse contexto, noções de diagnosticabilidade modular são apresentadas. Na arquitetura modular de Contant et al. (2006), duas hipóteses são consideradas: (i) não existem eventos não observáveis em comum entre os componentes, e (ii) o modelo do componente onde a falha está modelada deve ter a propriedade de *excitação persistente*, ou seja, para toda sequência de eventos de comprimento arbitrariamente longo gerada pela planta, a projeção nos eventos do componente também é formada por uma sequência arbitrariamente longa. Além dessas hipóteses limitarem a aplicação da arquitetura modular, a verificação da propriedade de excitação persistente não é apresentada nesse trabalho.

Recentemente, uma nova arquitetura para o diagnóstico de falhas, conhecida como diagnóstico síncrono, que também possui o intuito de evitar o uso do modelo global da planta para o diagnóstico foi proposta em Cabral e Moreira (2020). Nesse método, estimadores de estado dos modelos livres de falha dos componentes do sistema são implementados em paralelo. Assim, se um evento que não é possível em pelo menos uma das estimativas de estado locais é observado, a falha é diagnosticada. Apesar de evitar o crescimento computacional para o diagnóstico, na arquitetura síncrona, a linguagem livre de falha pode ser maior que a linguagem livre de falha do sistema quando existem eventos não observáveis em comum entre os componentes. Esse resultado pode acrescentar um atraso no diagnóstico ou até mesmo tornar um sistema que é diagnosticável de forma monolítica em não diagnosticável de forma síncrona. Em Cabral e Moreira (2020), arquiteturas síncronas centralizada e descentralizada são propostas e as noções de diagnosticabilidade e codiagnosticabilidade síncronas, bem como um método para verificar essas propriedades, são apresentados. Em Veras et al. (2021), esse método foi estendido para considerar uma arquitetura distribuída com o objetivo de se reduzir a linguagem livre de falha para o diagnóstico síncrono.

Neste trabalho, um método de diagnóstico síncrono descentralizado com coordenação (DSDC) que evita o crescimento da linguagem observável livre de falha para o diagnóstico síncrono é proposto. Para isso, um coordenador, que recebe a informação do comportamento não observável dos estimadores de estado locais em forma de *clusters*, é utilizado para reconstruir o comportamento livre de falha do sistema. Para tanto, um protocolo de comunicação entre estimadores locais e o coordenador, e um algoritmo para o processo de coordenação que informa o diagnóstico, são propostos. O método DSDC calcula o alcance não observável do comportamento livre de falha do modelo do sistema online após a observação de um evento. Em seguida, o coordenador verifica se o evento observado é viável na estimativa de estados atual do comportamento livre de falha. Se a resposta for não, a falha é diagnosticada. O coordenador é responsável por eliminar o excesso da linguagem livre de falha aceita pelo diagnóstico síncrono, garantindo a mesma eficiência da abordagem monolítica. Em geral, o custo computacional do método proposto é menor do que utilizar o modelo global do sistema para o diagnóstico. O ganho computacional em se utilizar o método DSDC ocorre principalmente em aplicações com maior número de eventos observáveis, o que é uma tendência na Indústria 4.0 que tem, como um dos pilares, o aumento de sensoriamento dos processos.

Este artigo está organizado da seguinte forma: na seção 2, a notação utilizada neste trabalho e a fundamentação teórica de diagnóstico de falhas e diagnóstico síncrono descentralizado de SEDs são apresentados. Na seção 3, o problema do diagnóstico síncrono descentralizado com coordenador é formulado e o método DSDC é introduzido. Na seção 4, o método DSDC é aplicado a um sistema de manufatura. As conclusões são apresentadas na seção 5.

2. FUNDAMENTAÇÃO TEÓRICA

Um autômato é denotado por $G = (Q, \Sigma, f, q_0)$, em que Q é o conjunto de estados, Σ é o conjunto finito de eventos, $f : Q \times \Sigma \to Q$ é a função de transição, e q_0 é o estado inicial. Uma transição em G é denotada por (q, σ, q'), em que $f(q, \sigma) = q'$. O domínio de f pode ser estendido para $Q \times \Sigma^\star$, em que $\Sigma^\star$ é o fecho de Kleene de Σ. A função $\Gamma_G : Q \to 2^\Sigma$ representa a função de eventos viáveis, em que $\Gamma_G(q)$ é o conjunto de todos os eventos σ tais que $f(q, \sigma)$ é definida. A função Γ_G também pode ser aplicada para um conjunto de estados $E \subseteq Q$, da seguinte forma: $\Gamma_G(E) = \cup_{q \in E} \Gamma_G(q)$. O símbolo ε é utilizado para representar a sequência vazia. A linguagem L gerada por G é dada por $L(G) := \{s \in \Sigma^\star : f(q_0, s) \text{ é definida}\}$. O prefixo-fechamento de L é definido como $\overline{L} = \{s \in \Sigma^\star : (\exists t \in \Sigma^\star)[st \in L]\}$. Uma linguagem $L \subseteq \Sigma^\star$ é dita ser viva se para todo $s \in L$, existe $\sigma \in \Sigma$ tal que $s\sigma \in L$. A parte acessível de G é denotada por $Ac(G) = (Q_{ac}, \Sigma, f_{ac}, q_0)$ (Cassandras e Lafortune, 2008).

O conjunto de eventos de G pode ser particionado como $\Sigma = \Sigma_o \dot\cup \Sigma_u$, em que Σ_o e Σ_u denotam os conjuntos de eventos observáveis e não observáveis, respectivamente. O alcance não observável de um estado $q \in Q$ é definido como $UR(q) = \{q' \in Q : (\exists t \in \Sigma_u^\star)[f(q, t) = q']\}$. A projeção

$P_s^l : \Sigma_l^\star \to \Sigma_s^\star$ e a projeção inversa $P_s^{l^{-1}} : \Sigma_s^\star \to 2^{\Sigma_l^\star}$, em que $\Sigma_s \subset \Sigma_l$ são definidas da forma usual (Cassandras e Lafortune, 2008). Sejam G_1 e G_2 dois autômatos, a composição paralela dos autômatos G_1 e G_2 é denotada como $G = G_1 \| G_2$ (Cassandras e Lafortune, 2008).

2.1 Diagnosticabilidade de sistemas a eventos discretos

Uma linguagem L é diagnosticável se sempre é possível detectar e isolar a ocorrência de eventos de falha em um número limitado de eventos gerados após a ocorrência da falha. O conjunto de eventos de falha é denotado como Σ_f, em que $\Sigma_f \subseteq \Sigma_u$. Neste trabalho, sem perda de generalidade, é suposto que existe somente um evento de falha, ou seja, $\Sigma_f = \{\sigma_f\}$. Uma sequência de falha é uma sequência de eventos $s \in L$ em que σ_f é um dos eventos que forma s. Por outro lado, sequências livres de falha não possuem σ_f.

A linguagem $L_N \subset L$ denota o conjunto de todas as sequências livres de falha de L. O conjunto de todas as sequências de falha de L é dado por $L_F = L \setminus L_N$, em que $\setminus$ denota a diferença de conjuntos. Os autômatos que geram as linguagens L_N e $\overline{L_F}$ são denotados por G_N e G_F, respectivamente. A definição de diagnosticabilidade de SEDs é apresentada a seguir.

Definição *1.* (Diagnosticabilidade). A linguagem viva e prefixo-fechada L é diagnosticável em relação à projeção $P_o : \Sigma^\star \to \Sigma_o^\star$ e Σ_f se

$(\exists z \in \mathbb{N})(\forall s \in L_F)(\forall st \in L_F, \|t\| \geq z \Rightarrow P_o(st) \notin P_o(L_N))$,

em que $\|.\|$ denota o comprimento de uma sequência. □

A definição 1 indica que L é diagnosticável se e somente se todas as sequências de falha $st \in L_F$ com comprimento arbitrariamente longo após a ocorrência da falha não possuem a mesma projeção P_o que qualquer sequência livre de falha $\omega \in L_N$.

2.2 Diagnóstico síncrono descentralizado de SEDs

No diagnóstico síncrono descentralizado (DSD) (Cabral e Moreira, 2020), o sistema G é formado por r componentes locais, tal que $G = \|_{i=1}^r G_i$. Diagnosticadores locais D_i são implementados para cada componente local $G_i = (Q_i, \Sigma_i, f_i, q_{0,i})$, que são calculados a partir do modelo do comportamento livre de falha de G_i, $G_{N_i} = (Q_{N_i}, \Sigma_i \setminus \Sigma_f, f_{N_i}, q_{0,i})$, $i = 1, \ldots, r$. O conjunto de eventos locais Σ_i é particionado em dois conjuntos: observável, $\Sigma_{i,o}$, e não observável, $\Sigma_{i,u}$, $\Sigma_i = \Sigma_{i,o} \dot\cup \Sigma_{i,u}$. É importante destacar que um evento pode ser observável para um diagnosticador local D_i e não observável para outro diagnosticador local D_j, para $i, j \in \{1, \ldots, r\}$ e $i \neq j$.

Em Cabral e Moreira (2020), é mostrado que a linguagem livre de falha para o DSD, L_{N_a}, pode ser maior que a linguagem livre de falha observável do sistema, $P_o(L_N)$. A linguagem L_{N_a} pode ser escrita como $L_{N_a} = \cap_{i=1}^r P_{i,o}^{o^{-1}}(P_{i,o}(L_{N_i}))$, em que $P_{i,o}^o : \Sigma_o^\star \to \Sigma_{i,o}^\star$ e $P_{i,o} : \Sigma^\star \to \Sigma_{i,o}^\star$ são projeções. A definição de codiagnosticabilidade síncrona é apresentada a seguir.

Definição *2.* (Codiagnosticabilidade síncrona). Seja $G_N = \|_{i=1}^r G_{N_i}$, em que G_{N_i} é o autômato que modela o comportamento livre de falha de G_i. Suponha que L_{N_i} denota a linguagem gerada por G_{N_i}, para $i = 1, \ldots, r$.

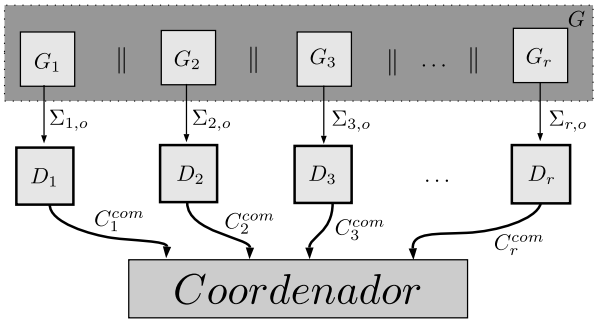


Figura 1. Arquitetura do método de diagnóstico síncrono descentralizado com coordenação.

Seja $P_o : \Sigma^\star \to \Sigma_o^\star$, em que $\Sigma_o = \cup_{i=1}^r \Sigma_{i,o}$. Logo, L é codiagnosticável de forma síncrona em relação a P_o, L_{N_i}, $i = 1, \ldots, r$, e Σ_f se

$(\exists z \in \mathbb{N})(\forall s \in L_F)(\forall st \in L_F, \|t\| \geq z \Rightarrow P_o(st) \notin L_{N_a})$.

□

Como $P_o(L_N) \subseteq L_{N_a}$ (Cabral e Moreira, 2020), um sistema pode ser diagnosticável de acordo com a definição 1 e não ser sincronamente codiagnosticável de acordo com a definição 2. O crescimento da linguagem livre de falha observável L_{N_a} para o método DSD está associado com a perda de sincronização de eventos não observáveis em comum entre dois ou mais componentes do sistema. Na sequência, o método DSDC que previne o crescimento da linguagem livre de falha para o diagnóstico síncrono é apresentado.

3. DIAGNÓSTICO SÍNCRONO DESCENTRALIZADO COM COORDENAÇÃO

Neste trabalho, um método de diagnóstico síncrono descentralizado com coordenação (DSDC) é proposto. O DSDC é baseado em estimadores locais D_i implementados em conjunto com um coordenador $\mathcal{C}$ que recebe informação de D_i, $i = 1, \ldots, r$. O coordenador $\mathcal{C}$ utiliza a estimativa de estado local de D_i para refinar o diagnóstico e informar se o evento de falha ocorreu. Cada componente G_i possui um local de medição, LM_i, que fornece a observação de eventos diretamente para o seu estimador de estados D_i. Assim como no método DSD, dois ou mais diagnosticadores locais podem ter eventos observáveis em comum.

No DSDC, quando um evento $\sigma_o \in \Sigma_{i,o}$ é observado por um local de medição de algum componente, LM_i, essa informação é enviada para o estimador D_i, que atualiza a estimativa de estado atual de G_{N_i}. Posteriormente, D_i envia ao coordenador um subautômato de G_{N_i}, referido neste artigo como *cluster* C_i^{com}, composto por: (i) os estados da última estimativa de estado em que σ_o é viável, (ii) os estados da estimativa de estado alcançada após a ocorrência de σ_o, e (iii) todas as transições não observáveis relacionadas a esses estados. Neste trabalho, a comunicação entre os agentes é suposta ser ideal, ou seja, não há atraso de comunicação e/ou perda de pacotes. A arquitetura do método DSDC é apresentada na figura 1.

Após os *clusters* C_i^{com} serem comunicados pelos estimadores de estado, a composição entre todos os *clusters* é realizada pelo coordenador com o objetivo de rastrear as

possíveis sincronizações dos eventos não observáveis. Isso é necessário para prevenir que o esquema de diagnóstico considere sequências livre de falha que não podem ser geradas pelo modelo global da planta. Essas sequências livres de falha excedentes, causadas pela perda de sincronização de eventos não observáveis, constituem o crescimento da linguagem livre de falha para o método DSD apresentado em Cabral e Moreira (2020).

3.1 Método DSDC

Para sincronizar corretamente os eventos não observáveis em comum entre os componentes no esquema DSDC, o coordenador $\mathcal{C}$ deve compor os *clusters* comunicados por D_i após a observação de um evento. Para tanto, um protocolo de comunicação entre D_i e $\mathcal{C}$, e um procedimento de diagnóstico que indica se a falha ocorreu após a observação de uma sequência de eventos, são propostos. Antes de apresentar o protocolo de comunicação, é necessário definir o autômato *cluster*. Dado um autômato $G = (Q, \Sigma = \Sigma_o \dot{\cup} \Sigma_u, f, q_0)$, um subconjunto de estados de Q, $E \subseteq Q$, e um evento observável σ_o, o autômato *cluster* de G é definido como $C(G, E, \sigma_o) = (Q_C, \Sigma, f_C, \emptyset)$, em que Q_C corresponde aos estados $q \in E$, tais que σ_o é viável, e todos os estados $UR(q')$, tais que (q, σ_o, q') existe em G. Além disso, $f_C(q, \sigma_o) = f(q, \sigma_o) = q'$, e $f_C(q'', \sigma_u) = f(q'', \sigma_u)$ para todo $q'' \in UR(q')$, sendo $\sigma_u \in \Sigma_u$. O protocolo de comunicação do método DSDC é dado pelo algoritmo 1.

O algoritmo 1 é inicializado nas linhas 1-5, em que cada estimador de estado local D_i, para $i = 1, \ldots, r$, envia um subautômato de G_{N_i}, $S_{0,i}$, que corresponde ao alcance não observável dos estados iniciais de G_{N_i} e as transições não observáveis associadas a esses estados. Em seguida, nas linhas 6-16, o procedimento de comunicação é detalhado quando um evento σ_o é observado. Se o evento observado $\sigma_o \in \Sigma_{i,o}$ é viável na estimativa de estado atual de D_i, E_i, o *cluster* $C_i^{com} = C_i(G_{N_i}, E_i, \sigma_o)$ é calculado e comunicado para $\mathcal{C}$. Caso contrário, $C_i^{com} = \emptyset$ é comunicado para o coordenador $\mathcal{C}$.

É importante ressaltar que o cálculo de $C_i(G_{N_i}, E_i, \sigma_o)$, necessário na linha 9 do algoritmo 1, pode ser feito linearmente utilizando qualquer algoritmo de busca de grafos (Cormen et al., 2009).

3.2 Procedimento de diagnóstico

No método DSDC, o coordenador $\mathcal{C}$ calcula uma composição entre r *clusters* a cada vez que um novo evento σ_o é observado, com o objetivo de verificar se σ_o é viável de acordo com o comportamento livre de falha do sistema. Cada novo *cluster* comunicado é incorporado em um outro previamente armazenado utilizando a operação de união de *clusters*, definida na sequência [1].

Definição *3.* (União de *clusters*). Sejam $C_1 = (Q_1, \Sigma_1, f_1, \emptyset)$ e $C_2 = (Q_2, \Sigma_2, f_2, \emptyset)$ dois *clusters*. A operação de união entre C_1 e C_2, é definida como $C = C_1 \sqcup C_2 = (Q, \Sigma, f, \emptyset)$, em que $Q = Q_1 \cup Q_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$, e $f : Q \times \Sigma \to Q$, em que $f(q, \sigma) = f_1(q_1, \sigma_1) = q_1'$ se $q_1, q_1' \in Q_1$ e $\sigma_1 \in \Sigma_1$; $f(q, \sigma) = f_2(q_2, \sigma_2) = q_2'$ se $q_2, q_2' \in Q_2$ e $\sigma_2 \in \Sigma_2$; caso contrário, $f(q, \sigma)$ é indefinida. □

[1] Esta operação é inspirada na operação de união de grafos apresentada em Harary (1969).

Algoritmo 1 Protocolo de comunicação do DSDC

Entrada: $G_{N_i} = (Q_{N_i}, \Sigma_i \setminus \Sigma_f, f_{N_i}, q_{0,i})$, para todo $i \in \{1, \ldots, r\}$.

1: **para todo** $i = 1, \ldots, r$ **faça**
2: Calcule $E_i \leftarrow UR(q_{0,i})$
3: Calcule o subautômato $S_{0,i} = (E_i, \Sigma_{i,u}, f_{S_i}, q_{0,i})$, em que $f_{S_i}(q, \sigma_u) = f_{N_i}(q, \sigma_u)$ para $q \in E_i$ e $\sigma_u \in \Sigma_{i,u}$
4: Envie $S_{0,i}$ para o Coordenador
5: **fim para todo**
6: Aguarde a observação de um evento $\sigma_o \in \Sigma_{i,o}$
7: **para todo** G_{N_i} em que $\sigma_o \in \Sigma_{i,o}$ **faça**
8: **se** $\sigma_o \in \Gamma_{G_{N_i}}(E_i)$ **faça**
9: Calcule o *cluster* $C_i^{com} \leftarrow C_i(G_{N_i}, E_i, \sigma_o)$ e envie para o Coordenador
10: Calcule $E_i' \leftarrow \cup_{q_i \in E_i} UR(f(q_i, \sigma_o))$
11: Atualize $E_i \leftarrow E_i'$
12: **senão**
13: Envie $C_i^{com} \leftarrow \emptyset$ para o Coordenador
14: **fim se**
15: **fim para todo**
16: Retorne para a linha 6

Em seguida, a composição síncrona de *clusters* é definida.

Definição *4.* (Composição síncrona de *clusters*). Sejam $C_1 = (Q_1, \Sigma_1, f_1, \emptyset)$ e $C_2 = (Q_2, \Sigma_2, f_2, \emptyset)$ dois *clusters*. A composição síncrona entre C_1 e C_2 é definida como $sync((C_1, C_2), R) = Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, f, R)$, em que $R \subseteq Q_1 \times Q_2$ é o conjunto de estados iniciais, $f((q_1, q_2), \sigma) = (f_1(q_1, \sigma), f_2(q_2, \sigma))$ se $\sigma \in \Gamma_{C_1}(q_1) \cap \Gamma_{C_2}(q_2)$; $f((q_1, q_2), \sigma) = (f_1(q_1, \sigma), q_2)$ se $\sigma \in \Gamma_{C_1}(q_1) \setminus \Sigma_2$; $f((q_1, q_2), \sigma) = (q_1, f_2(q_2, \sigma))$ se $\sigma \in \Gamma_{C_2}(q_2) \setminus \Sigma_1$; caso contrário, indefinida. □

Lema *1.* A definição 4 é equivalente à composição paralela entre autômatos usando R como o conjunto de estados iniciais.

Prova. A prova foi omitida por falta de espaço. ■

O coordenador utiliza a parte não observável dos *clusters* comunicados para reconstruir corretamente o comportamento livre de falha global do sistema. Isso é feito sincronizando-se os *clusters* de todos os componentes do sistema após a observação de um evento. No algoritmo 2, os passos para implementar o coordenador são apresentados.

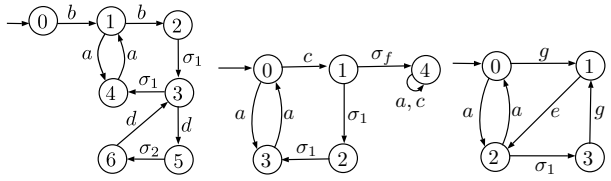
Exemplo *1.* Considere os autômatos G_i e seus respectivos modelos livres de falha G_{N_i}, $i = 1, 2, 3$, ilustrados nas figuras 2 e 3, respectivamente. Os conjuntos de eventos de G_1, G_2 e G_3 são, respectivamente, $\Sigma_1 = \{a, b, d, \sigma_1, \sigma_2\}$, $\Sigma_2 = \{a, c, \sigma_1, \sigma_f\}$, e $\Sigma_3 = \{a, e, g, \sigma_1\}$, em que $\Sigma_{1,o} = \{b, d\}$, $\Sigma_{2,o} = \{a, c\}$, e $\Sigma_{3,o} = \{a, e, g\}$, e $\Sigma_{1,u} = \{a, \sigma_1, \sigma_2\}$, $\Sigma_{2,u} = \{\sigma_1, \sigma_f\}$, e $\Sigma_{3,u} = \{\sigma_1\}$ são os conjuntos de eventos observáveis e não observáveis de G_1, G_2 e G_3. Portanto, o conjunto de eventos observáveis de G é $\Sigma_o = \Sigma_{1,o} \cup \Sigma_{2,o} \cup \Sigma_{3,o} = \{a, b, c, d, e, g\}$ e o conjunto de eventos de falha é $\Sigma_f = \{\sigma_f\}$.

Considere que a sequência de falha $s_f = bc\sigma_f a^n$, $n \in \mathbb{N}$, foi gerada pelo sistema. O procedimento de diagnóstico utilizando-se os algoritmos 1 e 2 é ilustrado como segue. Antes da observação de qualquer evento, o algoritmo 1 envia $S_{0,i} = (q_{0,i}, \Sigma_{i,u}, f_{0,i}, q_{0,i})$, $i = 1, 2, 3$, para o coordenador $\mathcal{C}$. Note que $S_{0,i}$, $i = 1, 2, 3$, corresponde somente aos estados iniciais de G_{N_i}, $i = 1, 2, 3$. Em

Algoritmo 2 Procedimento de diagnóstico

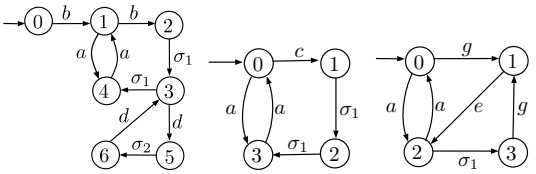
Entrada: $\Sigma_i = \Sigma_{i,o} \cup \Sigma_{i,u}$ for $i = 1, \ldots, r$, $\Sigma_o = \cup_{i=1}^{r} \Sigma_{i,o}$, subautômatos $S_{0,i}$, e os *clusters* comunicados C_i^{com}, para $i = 1, \ldots, r$.
Saída: Decisão do diagnóstico.

1: Defina $R \leftarrow \emptyset$ e $I \leftarrow \emptyset$
2: Defina $C_i' \leftarrow \emptyset$, para $i = 1, \ldots, r$
3: Após receber os subautômatos $S_{0,i}$, $i = 1, \ldots, r$, calcule $S = \|_{i=1}^{r} S_{0,i}$
4: Atribua a R o estado inicial de S
5: Defina $C_i \leftarrow S_{0,i}$, $i = 1, \ldots, r$
6: Aguarde a observação de um evento $\sigma_o \in \Sigma_{i,o}$
7: **para todo** C_i^{com} comunicado de D_i **faça**
8: **se** $C_i^{com} = \emptyset$ **faça**
9: Informe a ocorrência do evento de falha e pare
10: **senão**
11: Defina $C_i \leftarrow C_i \sqcup C_i^{com} = (Q_{C_i}, \Sigma_i, f_{C_i}, \emptyset)$
12: Defina $I \leftarrow I \cup \{i\}$
13: **fim se**
14: **fim para todo**
15: Calcule $S \leftarrow sync((C_1, C_2, \ldots, C_r), R) = (Q_S, \Sigma_N, f_S, R)$
16: **se** $f_S(q, \sigma_o)$ é indefinido para todo $q \in Q_S$ **faça**
17: Informe a ocorrência do evento de falha e pare
18: **senão**
19: Defina $R \leftarrow \{q' \in Q_S : f_S(q, \sigma_o) = q'\}$, para todo $q, q' \in Q_S$
20: **para todo** $i \in I$ **faça**
21: Calcule $C_i' = (Q_i', \Sigma_i, f_i', \emptyset)$ removendo de C_i todos os estados e transições que não alcançáveis após σ_o
22: Defina $C_i \leftarrow C_i'$
23: **fim para todo**
24: **fim se**
25: Defina $I \leftarrow \emptyset$
26: Retorne para a linha 6



(a) G_1 (b) G_2 (c) G_3

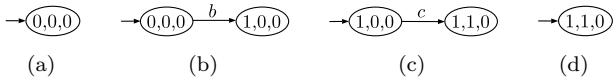
Figura 2. Autômatos G_1, G_2 e G_3.



(a) G_{N_1} (b) G_{N_2} (c) G_{N_3}

Figura 3. Autômatos G_{N_1}, G_{N_2} e G_{N_3}.

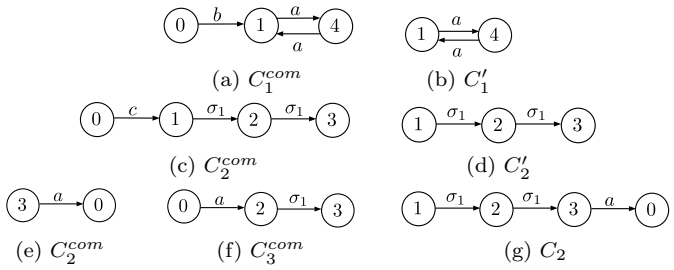
seguida, $S = S_{0,1} \| S_{0,2} \| S_{0,3}$, apresentado na figura 4(a), é calculado pelo algoritmo 2 que armazena o estado inicial de S em $R = \{(0,0,0)\}$. Nesse estágio, os algoritmos 1 e 2 aguardam a observação de um novo evento por D_1, D_2 ou D_3. Quando o evento b ocorre, ele é observado por D_1, e C_1^{com}, mostrado na figura 5(a), é comunicado. No algoritmo 2, a operação $C_1 \sqcup C_1^{com}$ é calculada na linha



(a) (b) (c) (d)

Figura 4. $S = \|_{i=1}^{3} S_{0,i}$ (a), S após a observação do evento b (b), S após a observação da sequência bc (c), e S após a observação da sequência bca (d).

11, resultando em C_1^{com}, que é atribuído a C_1. Visto que somente D_1 observa o evento b, o conjunto I é atualizado para $I = \{1\}$. De acordo com a linha 15 do algoritmo 2, um novo S é calculado, apresentado na figura 4(b). Visto que uma transição rotulada com b, nominalmente $((0,0,0), b, (1,0,0))$, existe no autômato S da figura 4(b), a falha não é detectada e o conjunto R é atualizado para os estados de S alcançados por transições rotuladas com b, *i.e.*, $R = \{(1,0,0)\}$ na linha 19 do algoritmo 2. Em seguida, C_1 é atualizado, ilustrado na figura 5(b), de acordo com as linhas 21-22 do algoritmo 2 removendo os estados e transições que não são alcançáveis após o evento b. Na sequência, o algoritmo 2 atualiza o conjunto $I = \emptyset$ e aguarda a observação de um novo evento.



(a) C_1^{com} (b) C_1' (c) C_2^{com} (d) C_2' (e) C_2^{com} (f) C_3^{com} (g) C_2

Figura 5. *Clusters* C_1^{com} (a) e C_1' (b) após a observação do evento b, C_2^{com} (c) e C_2' (d) após a observação da sequência bc, e C_2^{com} (e), C_3^{com} e C_2 após a observação da sequência bca.

O próximo evento gerado é c, observado pelo estimador de estados D_2 que envia C_2^{com}, apresentado na figura 5(c). Em seguida, C_2 é atualizado e, nesse caso, é igual a C_2^{com}. O conjunto I é atualizado para $I = \{2\}$ e um novo S é calculado, como mostra a figura 4(c). O conjunto R é atualizado para $R = \{(1,1,0)\}$ e um novo C_2 é calculado, como apresentado na figura 5(d). Quando o evento a ocorre, os estimadores de estados D_2 e D_3 enviam os *clusters* C_2^{com} e C_3^{com}, mostrados nas figuras 5(e) e 5(f), respectivamente. Os *clusters* C_2 (apresentado na figura 5(g)) e C_3 (ilustrado na figura 5(f)) são atualizados. Note que, C_3, nesse caso, é igual a C_3^{com}. O conjunto I é então calculado e é igual a $I = \{2,3\}$. Note também que $R = \{(1,1,0)\}$, que corresponde ao estado inicial de S. Além disso, os eventos a e σ_1 são comuns aos três componentes, porém não são viáveis em todos os três estados, ou seja, nenhuma transição deixa o estado $(1,1,0)$ de S. Portanto, S é igual ao autômato que possui somente o estado $(1,1,0)$, apresentado na figura 4(d), e visto que não há transições em S rotuladas com o evento a, a ocorrência do evento de falha é detectada na linha 17 do algoritmo 2. □

O teorema a seguir garante que a linguagem livre de falha considerada no método apresentado neste artigo é igual à linguagem livre de falha observável do sistema, $P_o(L_N)$.

Teorema *1.* Considere um sistema G, cuja linguagem gerada é L e a linguagem livre de falha é L_N. A linguagem

livre de falha observável considerada pelo método DSDC é igual à linguagem livre de falha observável do sistema $P_o(L_N)$, $P_o : \Sigma^\star \to \Sigma_o^\star$.

Prova. A prova foi omitida por falta de espaço. ■

O método DSDC pode ser visto como um meio de calcular, de forma online, o alcance não observável do modelo livre de falha do sistema, G_N, após a observação de uma sequência para verificar a viabilidade de um novo evento observado. Devido à composição realizada na linha 15 do algoritmo 2, a complexidade computacional do algoritmo 2 é, no cenário de pior caso, exponencial de acordo com o número de componentes do sistema. Entretanto, essa complexidade só é atingida se a maioria das transições dos modelos dos componentes são rotuladas com eventos não observáveis, visto que os *clusters* C_i^{com} comunicados são compostos por estados e transições não observáveis alcançados após a observação de um evento. Isso pode ser verificado no exemplo 1, em que G e G_N possuem 43 e 31 estados, respectivamente, enquanto que a soma dos estados de G_{N_1}, G_{N_2} e G_{N_3} é igual a 15 e o autômato S possui, no pior caso, 2 estados.

4. APLICAÇÃO EM UM SISTEMA DE MANUFATURA

Nesta seção, o método DSDC é aplicado a um sistema de manufatura instalado no Laboratório de Automação Industrial (LAI) da Universidade Federal de Santa Catarina (UFSC). Esse sistema é composto por três estações: (*i*) estação de distribuição, que transfere peças de trabalho para a próxima estação por meio de um braço robótico; (*ii*) estação de teste, que aceita ou rejeita a peça de trabalho de acordo com a sua altura; e (*iii*) estação de separação, que separa a peça de trabalho de acordo com o seu posicionamento.

O comportamento do sistema é descrito da seguinte forma. Na estação de distribuição, um cilindro pneumático empurra a peça de trabalho para fora do armazém. Em seguida, o braço robótico transfere a peça para a estação de teste em um elevador que verifica a altura da peça. Se a peça tem a altura adequada, um cilindro pneumático a empurra para a estação de separação. Caso contrário, a peça é descartada. Em seguida, na estação de separação, a posição da peça é verificada. Caso a peça esteja na posição correta, é transferida para a próxima estação. Caso contrário, a peça é descartada.

4.1 Modelo do sistema

Os modelos dos autômatos do comportamento livre de falha do sistema são ilustrados nas figuras 6(a), 6(b) e 6(c). Nesse sistema, o evento de falha σ_f modela o mau funcionamento da ventosa do braço robótico da estação de distribuição, responsável por travar a peça na posição por sucção enquanto é movimentada, apresentado na figura 6(d). Quando a falha ocorre, apesar da estação de distribuição realizar todo seu funcionamento cíclico, novas peças não são mais entregues para a estação de teste. Os autômatos G_2 e G_3 são iguais a G_{N_2} e G_{N_3}, respectivamente. É importante destacar que a ocorrência do evento de falha não leva o sistema a um travamento e

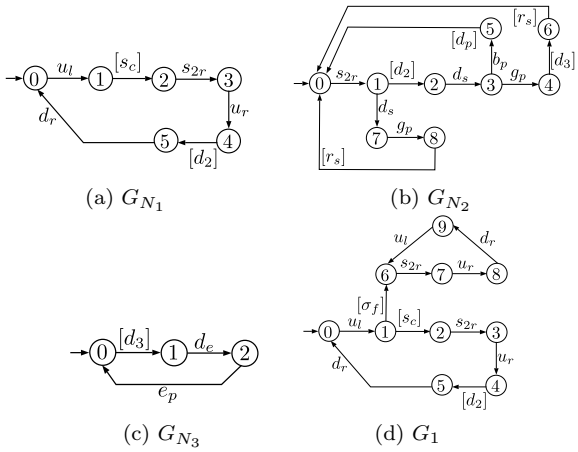


Figura 6. Comportamento livre de falha das estações: (a) Estação de distribuição - G_{N_1}, (b) Estação de teste - G_{N_2}, (c) Estação de separação - G_{N_3} e (d) Comportamento com a falha modelada na bancada de distribuição - G_1.

Tabela 1. Estados de G_1.

Estado	Significado
0	Peça na posição correta para transporte
1	Braço robótico na posição correta para ligar o vácuo
2	Peça travada pelo braço robótico com vácuo ligado
3	Braço robótico se movendo para entregar a peça
4	Braço robótico pronto para entregar a peça
5	Peça entregue com sucesso
6	Peça não foi travada e vácuo desligado
7	Braço robótico se movimentando sem peça
8	Braço robótico na posição de entrega sem peça
9	Braço robótico retornando para a posição neutra

Tabela 2. Estados de G_2.

Estado	Significado
0	Estação esperando uma nova peça
1	Estação pronta para testar uma nova peça
2	Peça entregue ao elevador
3	Altura da peça testada pelo sensor
4	Peça entregue e elevador na posição de cima
5	Cilindro pneumático descartou a peça
6	Estação esperando para voltar ao estado inicial
7	Teste de altura realizado sem peça
8	Elevador na posição de cima sem peça

que não é possível diagnosticar a falha utilizando apenas o autômato G_1.

O conjunto de eventos de G_1, G_2, e G_3 são $\Sigma_1 = \{u_l, s_c, s_{2r}, u_r, d_2, d_r, \sigma_f\}$, $\Sigma_2 = \{s_{2r}, d_2, d_s, g_p, r_s, b_p, d_3, d_p\}$, e $\Sigma_3 = \{d_3, d_e, e_p\}$, respectivamente, em que os conjuntos de eventos observáveis de cada componente são $\Sigma_{1,o} = \{u_l, s_{2r}, u_r, d_r\}$, $\Sigma_{2,o} = \{s_{2r}, d_s, g_p, b_p\}$ e $\Sigma_{3,o} = \{d_e, e_p\}$. Portanto, o conjunto de eventos de $G = G_1 \| G_2 \| G_3$ é $\Sigma = \{u_l, s_c, s_{2r}, u_r, d_2, d_r, \sigma_f, d_s, g_p, r_s, b_p, d_3, d_p, d_e, e_p\}$, em que o conjunto de eventos observáveis de G é $\Sigma_o = \Sigma_{1,o} \cup \Sigma_{2,o} \cup \Sigma_{3,o} = \{u_l, s_{2r}, u_r, d_r, d_s, g_p, b_p, d_e, e_p\}$ e o conjunto de eventos não observáveis de G é $\Sigma_u = \Sigma \backslash \Sigma_o$. Os estados de G_1, G_2 e G_3 e o significado dos eventos de Σ são apresentados nas tabelas 1, 2, 3 e 4. Note que os eventos não observáveis dos autômatos apresentados na figura 6 estão representados entre colchetes.

Tabela 3. Estados de G_3.

Estado	Significado
0	Estação esperando nova peça
1	Peça na posição correta para ser verificada
2	Peça detectada pelo sensor

Tabela 4. Eventos de G_1, G_2, e G_3.

Evento	Significado
u_l	Braço robótico sai da posição neutra
s_c	A sucção é ligada
s_{2r}	Estação de teste está pronta para receber peça
d_2	Peça é entregue ao elevador
u_r	Borda de subida da posição direita do braço
d_r	Borda de descida da posição direita do braço
σ_f	Ocorre a falha na sucção
d_s	Braço robótico retorna à posição segura
g_p	Cilindro pneumático empurra a peça para a esteira
d_3	Peça é entregue à estação de separação
r_s	Sistema retorna para o estado inicial
d_p	Peça é descartada
b_p	Cilindro pneumático empurra a peça para descarte
d_e	Peça é detectada
e_p	Peça é verificada

Os algoritmos 1 e 2 foram implementados com o objetivo de se detectar a falha apresentada no sistema de sucção da estação de distribuição. A falha foi detectada de forma bem sucedida depois do segundo ciclo de funcionamento da estação de teste após a ocorrência do evento de falha. É importante destacar que o modelo completo livre de falha do sistema $G_N = G_{N_1} \| G_{N_2} \| G_{N_3}$ tem 96 estados, enquanto que a soma de estados de G_{N_1}, G_{N_2} e G_{N_3} é igual a 26 e, no pior caso, o número total de estados do autômato S obtido no algoritmo 2 é igual a 9. Isso mostra a vantagem computacional de se usar o método DSDC para um sistema real, que geralmente é obtido a partir da composição de múltiplos componentes.

5. CONCLUSÃO

Neste trabalho, o método de diagnóstico síncrono descentralizado com coordenação é apresentado. O método é baseado em estimadores de estado locais do comportamento livre de falha dos componentes do sistema que comunicam *clusters* para um coordenador. O coordenador, por sua vez, realiza o diagnóstico com base no cálculo online da estimativa de estados do comportamento livre de falha do sistema global. A falha é diagnosticada quando o coordenador verifica que um evento observado não é viável na estimativa de estados atual. A eficiência do método apresentado neste trabalho é equivalente à abordagem clássica de diagnóstico monolítico, e em geral, possui menor custo computacional que a implementação do modelo global do sistema. Além disso, um exemplo prático é apresentado com o objetivo de ilustrar a aplicação do método e a redução do custo computacional do processo.

REFERÊNCIAS

Cabral, F.G. e Moreira, M.V. (2020). Synchronous Diagnosis of Discrete-Event Systems. *IEEE Transactions on Automation Science and Engineering*, 17(2), 921–932.

Cabral, F.G., Moreira, M.V., Diene, O., e Basilio, J.C. (2015). A Petri Net Diagnoser for Discrete Event Systems Modeled by Finite State Automata. *IEEE Transactions on Automatic Control*, 60(1), 59–71.

Carvalho, L.K., Basilio, J.C., e Moreira, M.V. (2012). Robust diagnosis of discrete event systems against intermittent loss of observations. *Automatica*, 48(9), 2068–2078.

Cassandras, C.G. e Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Spring, 2 edition.

Contant, O., Lafortune, S., e Teneketzis, D. (2006). Diagnosability of discrete event systems with modular structure. *Discrete Event Dynamic Systems: Theory and Applications*, 16(1), 9–37.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., e Stein, C. (2009). *Introduction to algorithms*. MIT Press, 3 edition.

Debouk, R., Lafortune, S., e Teneketzis, D. (2000). Coordinated decentralized protocols for failure diagnosis of discrete event systems. *Discrete Event Dynamic Systems: Theory and Applications*, 10(1), 33–86.

Debouk, R., Malik, R., e Brandin, B. (2002). A modular architecture for diagnosis of discrete event systems. *Proceedings of the IEEE Conference on Decision and Control*, 417–422.

Harary, F. (1969). *Graph Theory*. Addison-Wesley.

Keroglou, C. e Hadjicostis, C.N. (2018). Distributed Fault Diagnosis in Discrete Event Systems via Set Intersection Refinements. *IEEE Transactions on Automatic Control*, 63(10), 3601–3607.

Lefebvre, D. e Delherm, C. (2007). Diagnosis of des with Petri net models. *IEEE Transactions on Automation Science and Engineering*, 4(1), 114–118.

Moreira, M.V., Basilio, J.C., e Cabral, F.G. (2016). "polynomial time verification of decentralized diagnosability of discrete event systems" versus "decentralized failure diagnosis of discrete event systems": A critical appraisal. *IEEE Transactions on Automatic Control*, 61(1), 178–181.

Moreira, M.V., Jesus, T.C., e Basilio, J.C. (2011). Polynomial time verification of decentralized diagnosability of discrete event systems. *IEEE Transactions on Automatic Control*, 56(7), 1679–1684.

Qiu, W. e Kumar, R. (2005). Distributed diagnosis under bounded-delay communication of immediately forwarded local observations. *Proceedings of the American Control Conference*, 2, 1027–1032.

Qiu, W. e Kumar, R. (2006). Decentralized failure diagnosis of discrete event systems. *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, 36(2), 384–395.

Sampath, M., Sinnamohideen, K., Lafortune, S., e Teneketzis, D. (1995). Diagnosability of Discrete-Event Systems. *IEEE Transactions on Automatic Control*, 40(9), 1555–1575.

Veras, M.Z., Cabral, F.G., e Moreira, M.V. (2021). Distributed synchronous diagnosis of discrete event systems modeled as automata. *Control Engineering Practice*, 115, 104892.

Wang, Y., Yoo, T.S., e Lafortune, S. (2007). Diagnosis of discrete event systems using decentralized architectures. *Discrete Event Dynamic Systems: Theory and Applications*, 17(2), 233–263.