BUSCA HEURÍSTICA DE SUPERVISORES APROXIMADOS POR DISTINGUIDORES

Raquel S. S. Aguiar*, Antonio E. C. Cunha*, José E. R. Cury†, Max H. Queiroz†

*Praça General Tibúrcio, 80 - Praia Vermelha
Instituto Militar de Engenharia - Seção de Ensino em Engenharia Elétrica
Rio de Janeiro, RJ, Brasil

†Universidade Federal de Santa Catarina - Centro Tecnológico- Departamento de Automação e Sistemas
Florianópolis, SC, Brasil

Emails: raquelaguiar@ime.eb.br, carrilho@ime.eb.br, cury@das.ufsc.br, max@das.ufsc.br

Abstract— The use of distinguishers was introduced for Supervisory Control Theory in order to simplify the task of modeling specifications, while guaranteeing the synthesis of maximally permissive supervisors. On the other hand, by approximating the language of a distinguisher, a supervisor can be obtained with computational savings in the synthesis, although there is no guarantee that this is the maximally permissive solution. The main purpose of this work is to propose a procedure to obtain the least restrictive achievable supervisor, in the context of approximations in the Supervisory Control Theory with distinguishers. The procedure consists of a heuristic search for the supervisor with the highest language measure in the space formed by supervisors obtained using approximated distinguishers. A search procedure based in Genetic Algorithms was implemented and a case study illustrates the results of the method.

Keywords— Discrete Event-Systems, Supervisory Control Theory, Distinguishers, Metaheuristics

Resumo— O uso de distinguidores foi introduzido na Teoria de Controle Supervisório visando a simplificar a tarefa de modelagem das especificações ao mesmo tempo em que assegurava a síntese do supervisor maximamente permissivo. Além disso, ao aproximar-se a linguagem do distinguidor, é possível sintetizar um supervisor com menos esforço computacional, embora não haja garantia de que este supervisor seja a solução maximamente permissiva. O objetivo deste trabalho é propor um método para se obter o supervisor menos restritivo possível, dentre os obtidos por distinguidores aproximados na Teoria do Controle Supervisório. O método consiste na busca heurística pelo supervisor com a maior medida de linguagem no espaço formado por todos os supervisores sintetizados a partir de distinguidores aproximados. Para o presente trabalho, o método foi implementado utilizando-se o algoritmo genético como heurística e um estudo de caso ilustra os ganhos do método.

Palavras-chave— Sistemas a eventos discretos, Teoria de Controle Supervisório, Distinguidores, Metaheurísticas.

1 Introdução

A Teoria de Controle Supervisório (TCS) introduzida por RAMADGE e WONHAM (1989) provê um método formal para se obter agentes de controle para Sistemas a Eventos Discretos (SED), representados por supervisores. Porém, a modelagem de especificações complexas pode ser um obstáculo à aplicação deste método.

Com o objetivo de simplificar este passo da TCS, o conceito de distinguidores foi introduzido por (BOUZON et al., 2008). O distinguidor mapeia eventos do alfabeto original para um novo alfabeto refinado. Ao fazer uso da informação implícita nos refinamentos, os modelos para especificações complexas podem ser simplificados quando construídos no alfabeto refinado. Além disso, o supervisor obtido com o uso de distinguidores é tão permissivo quanto o obtido no alfabeto original, se a modelagem das especificações puder ser executada.

Outro obstáculo na aplicação da TCS é a complexidade da síntese destes supervisores, como por exemplo nos casos em que há explosão combinatória do número de estados do sistema devido a composição de vários subsistemas e requerimentos de funcionamento. Para reduzir esta explosão combinatória, abordagens como o Controle Modular (WONHAM e RAMADGE, 1988) e o Controle Modular Local (QUEIROZ e CURY, 2002) foram propostas.

No que concerne a este problema, o uso de distinguidores permite explorar o uso de aproximações do modelo exato do sistema visando a reduzir a complexidade da síntese do supervisor (CURY et al., 2013; TEIXEIRA et al., 2011). O supervisor resultante, no entanto, pode ser mais restritivo do que o ótimo e pode mesmo nem ser uma solução para o problema de controle, uma vez que pode conduzir o sistema a estados de bloqueio. As aproximações são de grande importância nos casos em que devido à grande complexidade do sistema, o supervisor não pode ser obtido com o distinguidor não-aproximado. Como alternativa, o supervisor mais permissivo e não-conflitante deve ser encontrado dentre todas as aproximações possíveis, mas uma busca exaustiva seria muito ineficiente.

Nesse contexto, este artigo sugere o emprego de busca heurística para explorar o espaço formado pelos supervisores obtidos com o uso de distinguidores aproximados. Será mostrado como a heurística pode aprimorar a eficiência da busca. Também será proposta, após uma pequena adaptação para o uso com distinguidores, a medida de linguagem introduzida por WANG e RAY (2002) para comparar os supervisores. O método de busca será então aplicado a uma linha de manufatura cuja solução de controle se beneficia do uso de distinguidores e suas aproximações.

O restante deste documento está organizado como se segue: a seção 2 faz uma breve introdução aos conceitos básicos da TCS e da TCS com distinguidores; a seção 3 descreve o método proposto para a busca heurística do espaço formado pelas aproximações; a seção 4 apresenta o estudo de caso e a seção5 expões as conclusões deste artigo.

2 Controle Supervisório com distinguidores

2.1 Conceitos preliminares

Este artigo está inserido no contexto de RAMADGE e WONHAM (1989) e ao leitor é indicado consultar CASSANDRAS e LAFORTUNE (2008) para uma introdução mais detalhada sobre o assunto.

O comportamento de SED é modelado por cadeias de eventos pertencentes a um alfabeto finito Σ. Σ^* é o con-

junto de todas as cadeias finitas de eventos de Σ, mais a cadeia vazia ϵ. A concatenação das cadeias s e u é denotada como su e a cadeia s é um prefixo de t, denotado $s \leq t$,se existe uma cadeia u tal que $su = t$. Uma linguagem é definida como qualquer subconjunto de Σ^*. O prefixo-fechamento de uma linguagem L é o conjunto $\overline{L} = \{s \in \Sigma^* \mid s \leq t, \textit{ para algum } t \in L\}$, de todos os prefixos de L. Uma linguagem L é prefixo-fechada se $L = \overline{L}$ e duas linguagens, K_1 e K_2, são não-conflitantes entre si se $\overline{K_1 \cap K_2} = \overline{K_1} \cap \overline{K_2}$.

Um autômato é definido por uma quíntupla $G = (Q, \Sigma, \delta, q_i, Q_m)$, em que Q é um conjunto de estados, Q_m é o conjunto de estados marcados, q_i é o estado inicial, Σ é o alfabeto, e $\delta : Q \times \Sigma \to Q$ é a função de transição parcial. O autômato G representa duas linguagens, a linguagem gerada $L(G) = \{s \in \Sigma^* \mid \hat{\delta}(q_i, s) \; definida\}$, em que $\hat{\delta}$ é a extensão δ para cadeias em Σ^*, e a linguagem marcada $L_m(G) = s \in \Sigma^* \mid \hat{\delta}(q_i, s) \in Q_m\}$. Um o autômato G que corresponde a um SED, $L(G)$ representa o conjunto de todas as cadeias que o SED pode gerar e $L_m(G)$ contém todas as cadeias que definem tarefas completas no sistema.

Para sistemas compostos, onde cada componente é modelado por um autômato G_i, $i = 1...n$, o autômato do sistema global é obtido pela composição síncrona $G = \|_{i=1}^{n} G_i$, obtida pela evolução paralela dos n autômatos, em que um evento comum a múltiplos autômatos só pode ocorrer se ocorre simultaneamente em todos eles. Pode ser mostrado que a linguagem do autômato resultante é o produto síncrono das linguagens de casa componente.

2.2 Controle Supervisório

Na Teoria de Controle Supervisório (TCS) (RAMADGE e WONHAM, 1989), o SED, planta, é modelado pelo autômato G e o alfabeto de eventos Σ é particionado no conjunto de eventos não-controláveis Σ_u, cujas ocorrências não podem ser inibidas, e no conjunto de eventos controláveis Σ_c, cujas ocorrências podem ser desabilitadas.

Um supervisor S pode ser representado por um autômato Z cuja transição de estados é dirigida pela ocorrência de eventos na planta G. A ação de controle de S, definida para cada estado do autômato correspondente Z, é desabilitar em G os eventos que não podem ocorrer em Z depois de uma dada sequência observada de eventos. O sistema em malha fechada S/G pode ser descrito pela composição síncrona $Z\|G$, que define as duas linguagens de malha fechada $L(S/G) = L(Z\|G)$ e $Lm(S/G) = L_m(Z\|G)$. Além de restringir o comportamento da planta ao desabilitar eventos controláveis, uma tarefa só é considerada completa em malha fechada se é marcada tanto pela planta quanto pelo supervisor. Um supervisor S é não-bloqueante para G se $\overline{L_m(S/G)} = L(S/G)$.

Dada a planta G e uma especificação genérica de linguagem $E \subseteq \Sigma^*$, a condição suficiente e necessária para a existência de um supervisor não-bloqueante S para G tal que $L_m(S/G) = K = L_m(G)\|E$ é que K deva ser controlável em relação a (e.r.a.) $L(G)$ e Σ_u, definido por $\overline{K}\Sigma_u \cap L(G) \subseteq \overline{K}$. Se K é não controlável, então deve ser computada a linguagem $SupC(K, G)$, o elemento supremo do conjunto $C(K, G)$ formado por todas as sublinguagens de K que são controláveis e.r.a. $L(G)$ e Σ_u. $SupC(K, G)$ é o comportamento mais permissivo possível de ser implementado por um supervisor não-bloqueante S para a planta G que atende a especificação K.

2.3 Distinguidores

A tarefa de modelagem de algumas especificações pode se tornar muito complexa e os distinguidores podem ser uma alternativa para simplificá-la (BOUZON et al., 2008; CURY et al., 2013). Os eventos podem ser refinados e a informação implícita carregada por estes refinamentos são usados para simplificar a modelagem de especificações. Para cada evento $\sigma \in \Sigma$, haverá um conjunto de refinamentos, denotados por Δ^σ, que herdam o atributo de controlabilidade do evento σ. O evento σ é dito ser a máscara para os eventos em Δ^σ. A união dos conjuntos de refinamentos para todo $\sigma \in \Sigma$ é o alfabeto refinado $\Delta = \dot{\bigcup}_{\sigma \in \Sigma} \Delta^\sigma$.

Considere um mapeamento mascarador $\Pi : \Delta^* \to \Sigma^*$ tal que $\Pi(\epsilon) = \epsilon$ e $\Pi(t\rho) = \Pi(t)\sigma$ para t $\in \Delta^*$, $\rho \in \Delta^\sigma$ e $\sigma \in \Sigma$ que mapeia cada cadeia em Δ^* para Σ^*. Este mapeamento pode ser aplicado a uma linguagem $L_d \subseteq \Delta^*$ tal que $\Pi(L_d) = \{s \in \Sigma \mid \exists t \in L_d, \Pi(t) = s\}$. Analogamente, um mapeamento reverso $\Pi^{-1} : \Sigma^* \to 2^{\Delta^*}$ é definido como $\Pi^{-1}(s) = \{t \in \Delta^* \mid \Pi(t) = s\}$ e também é aplicado a linguagens, como $\Pi^{-1}(L) = \{t \in \Delta^* \mid \Pi(t) \in L\}$.

O distinguidor é então um mapeamento $D : \Sigma^* \to 2^{\Delta^*}$ definido como se segue, para todo $s \in \Sigma^*$, $\sigma \in \Sigma$, $t \in \Delta^*$ e $\rho \in \Delta$:

$$\begin{array}{l} 1) D(\epsilon) = \{\epsilon\}\,; \\ 2) t\rho \in D(s\sigma) \Rightarrow t \in D(s) \textit{ e } \rho \in \Delta^\sigma; \\ 3) t \in D(s) \Rightarrow \exists \rho' \in \Delta^\sigma \textit{ tal que } t\rho' \in D(s\sigma). \end{array} \quad (1)$$

O distinguidor pode ser aplicado a linguagens, de forma que $\underline{D(L)} = \{t \in \Delta^* \mid \exists s \in L, t \in D(s)\}$ e $\Pi(D(L)) = L$ e $\overline{D(L)} = D(\overline{L})$. Se definido que $D(\Sigma^*) = L_D$, então $D(L) = \Pi^{-1}(L) \cap L_D$.

Se um distinguidor associa, para cada cadeia $s \in \Sigma^*$, uma, e somente uma, cadeia $t \in \Delta^*$, este distinguidor é dito ser preditivo. Neste texto, a linguagem de um distinguidor preditivo será denotado por L_D. Além disso para um distinguidor preditivo D, $D(\Pi(L_D)) = L_D$. O autômato H, tal que $L_m(H) = L_D$, pode ser construído modularmente, em que cada módulo distingue as instâncias de uma máscara em particular.

O teorema 1, apresentado em CURY et al. (2013), demonstra como usar o distinguidor para sintetizar um supervisor e enuncia a equivalência entre o supervisor obtido por distinguidores preditivos e o obtido no alfabeto original Σ.

Teorema 1 *Dada a planta G, a especificação genérica $E \subset \Sigma^*$ e o distinguidor preditivo D, seja G_d um autômato tal que $L_{(m)}(G_d) = D(L_{(m)}(G)) = L_{(m)}(G) \cap L_D$ e seja E_d a especificação genérica modelada em Δ^* expressando os mesmos requerimentos que E. Então para $K_d = E_d \cap L_m(G_d) = D(K) = D(E \cap L_m(G))$:*

$$\Pi(supC(K_d, G_d)) = supC(K, G) \; e \quad (2)$$

$$supC(K_d, G_d) = D(supC(K, G)). \quad (3)$$

A complexidade da síntese de um supervisor utilizando-se distinguidores preditivos é igual a da síntese na TCS original. A razão para tal é que enquanto o uso de distinguidores diminui a complexidade da especificação, ele aumenta a complexidade da planta, uma vez que a planta passa a conter o modelo do distinguidor.

2.4 Aproximações na TCS com distinguidores

Com o objetivo de poupar esforços computacionais por meio da redução da complexidade envolvida na síntese de supervisores, CURY et al. (2013) e TEIXEIRA et al. (2011) introduziram a síntese de supervisores por distinguidores aproximados.

Seja D_{aj} e D distinguidores não-preditivo e preditivo, respectivamente, e que $L_D \subseteq L_{D_{aj}} \subseteq \Delta^*$. D_{aj} é dito ser uma aproximação para o distinguidor D e pode relacionar mais de um refinamento para cada máscara em Σ. O autômato H_{aj} para um distinguidor aproximado pode ser construído ao se compor alguns módulos de H, mas não todos. As linguagens gerada e marcada para uma aproximação G_{aj} para a planta G_d são tais que $L(G_{aj}) = \Pi^{-1}(L(G)) \cap L_{D_{aj}}$ e $L_m(G_{aj}) = \Pi^{-1}(L_m(G)) \cap L_{D_{aj}}$.

Dadas duas linguagens $L_1 \subseteq L_2 \subseteq \Delta^*$ e um conjunto de máscaras $\Lambda \subseteq \Sigma$, $\underline{L_1}$ é dita ser Λ-preservante e.r.a L_2 se $\forall \sigma \in \Lambda$ e $t \in \overline{L_1}$ é verdadeiro que $t\Delta^\sigma \cap \overline{L_2} \neq \emptyset \Rightarrow t\Delta^\sigma \cap \overline{L_1} \neq \emptyset$. O conjunto $P_\Lambda(L_1, L_2) = \{K \subseteq L_1 \mid K \text{ é } \Lambda\text{-preservante e.r.a. } L_2\}$ é o conjunto de todas as sublinguagens de L_1 e Λ-preservantes e.r.a. L_2. Pode ser demonstrado que $P_\Lambda(L_1, L_2)$ é não-vazia e fechada para a união, além disso, o elemento supremo desse conjunto é denotado por $SupP_\Lambda(L_1, L_2)$ (CURY et al., 2013).

Dada uma aproximação G_{aj} para a planta G_d, conforme definido acima, e uma especificação genérica$E_d \subseteq \Delta^*$, para $K_{aj} = E_d \cap L_m(G_{aj})$, se $supC(K_{aj}, G_{aj})$ e L_D são não-conflitantes, um supervisor S_{aj} que implementa $supC(K_{aj}, G_{aj}) \cap L_D$ é um supervisor não-bloqueante para o sistema. Adicionalmente, $supC(K_{aj}, G_{aj}) \cap L_D \subseteq supC(K_d, G_d) \subseteq supP_{\Sigma_u}(K_{aj}, G_{aj}) \cap L_D$. Isto é, o supervisor obtido por distinguidor aproximado, se é não-conflitante com L_D, é tão ou mais permissivo que o supervisor obtido com o distinguidor preditivo.

Se $supC(K_{aj}, G_{aj})$ e L_D são não-conflitantes, uma condição suficiente, mas não necessária, para otimidade de S_{aj} é que $supC(K_{aj}, G_{aj}) = supP\Sigma_u(K_{aj}, G_{aj})$, em que $supP_\Lambda(K_{aj}, G_{aj})$, para $\Lambda = \Sigma_u$.

A complexidade da síntese de um supervisor por distinguidor aproximado é $O(ng^2 \cdot ne^2 \cdot nha^2)$ (CASSANDRAS e LAFORTUNE, 2008), em que ng, ne e nha são, respectivamente, o número de estados dos autômatos para a planta distinguida por Δ^*, a especificação genérica no alfabeto refinado e o distinguidor aproximado. Uma vez que o autômato para o distinguidor aproximado H_{aj} é construído pela composição de diferentes módulos, o número de estados de H_{aj} potencialmente cresce exponencialmente com a adição de novos módulos, sendo limitado pelo tamanho potencial de H, que contém todos os módulos distinguidores. Como consequência, as sínteses de supervisores por distinguidores aproximados são potencialmente menos complexas que as baseadas nos distinguidores preditivos, sendo a complexidade exponencialmente dependente do número de módulos distinguidores incluídos na aproximação.

3 Metodologia

Inúmeras aproximações podem ser construídas para um distinguidor, logo haverá inúmeros supervisores possíveis, um para cada aproximação. Dado que a informação provida pelo distinguidor influencia em quão permissivo é o supervisor obtido pelo mesmo, deve ser esperado que aproximações diferentes resultem em diferentes graus de permissividade dos supervisores. Explorar todo o espaço formado por estes supervisores por meio da busca por força bruta é ineficiente e a heurística pode elevar a eficiência desta tarefa (PARDALOS et al., 2011). Para se embutir heurísticas na busca, o espaço deverá ser totalmente ordenado por algum operador. Com este propósito, este artigo sugere a medida de linguagem (WANG e RAY, 2002; SURANA e RAY, 2004), apresentada a seguir.

3.1 Medida de linguagem na TCS-D

A medida de linguagem, introduzida em WANG e RAY (2002), foi proposta para uma avaliação quantitativa do desempenho de um supervisor para SED. Além disso, a medida também provê um ordenamento total para um conjunto de linguagens, sendo ideal para comparar supervisores diferentes, problema central de métodos heurísticos para busca em espaço formado por distinguidores.

Seja $G = (Q, \Sigma, \delta, q_i, Q_m)$ um autômato. A função característica $\chi : Q \to [-1, 1]$ é definida, para $q \in Q$, como (i) $\chi(q) \in [-1, 1] - \{0\}$, se $q \in Q_m$, e (ii) $\chi(q) = 0$ se $q \notin Q_m$. A função característica atribui a cada estado de G um peso de valor real de acordo com como este estado afeta o comportamento do SED. Para o presente trabalho, a função característica atribui o peso de valor 1 para todos os estados marcados, uma vez que todos estes impactam o comportamento da mesma forma.

O custo de evento $\widetilde{\psi} : \Sigma \times Q \to [0, 1)$ é definido, para $q_j \in Q$ e $\sigma_k \in \Sigma$, como (i) $\widetilde{\psi}[\sigma_k, q_j] \equiv \widetilde{\psi}_{jk} \in [0, 1)$; (ii) $\sum_{j=1}^{m} \widetilde{\psi}_{jk} < 1$; e (iii) $\widetilde{\psi}[\sigma_k, q_j] = 0$, se $\delta(q_j, \sigma_k)$ é não-definida. O conceito de custo de evento é similar à probabilidade condicional, pois o custo de evento em particular depende da sequência de eventos que o precede. Para este trabalho, todos os eventos definidos para um mesmo estado são considerados equiprováveis e ao custo do evento é atribuído o valor de 0,8 dividido pelo número de eventos cuja ocorrência é definida nesse estado.

O custo de cadeia $\widetilde{\psi} : \Sigma^* \times Q \to [0, 1]$ é definido, para $q_j \in Q$, $s \in \Sigma^*$ e $\sigma \in \Sigma$, como (i) $\widetilde{\psi}[\epsilon, q_k] = 1$, e (ii) $\widetilde{\psi}[\sigma s, q_j] = \widetilde{\psi}[\sigma, q_j]\widetilde{\psi}[s, \delta(q_j, \sigma)]$.

O custo de cadeia é o produto cumulativo do custo de evento de todos os eventos da cadeia. Com a finalidade de garantir que o custo de cadeia convergirá para cadeias de comprimento infinto, o custo de evento é limitado a $[0, 1)$ como acima.

Por fim, a medida de linguagem, $\mu : L(G) \to \Re$, é a soma do custo de cadeia de todas as cadeias contidas em G, ponderadas pelo peso atribuído pela função característica ao estado no qual a cadeia termina, ou seja, $\mu(L(G)) = \sum_{q_j \in Q} \upsilon(L(q_i, q_j))\chi(q_j)$.

Um método algébrico para computar a medida de linguagem de $L(G)$ foi proposto em (WANG e RAY, 2002). Primeiramente, um custo de transição de estado $\psi : Q \times Q \to [0, 1)$ é definido como a soma do custo de evento de todos os eventos cuja ocorrência implica na transição de dado um estado para um outro em particular.

O custo de transição de estado gera a matriz $n \times n$ de custo de transição de estado Ψ. A medida de linguagem para $L(G)$ é então o $i - simo$ elemento do vetor $\overline{\mu} \equiv [\mu_1 \, \mu_2 \ldots \mu_n]^T$, em que i é o índice do estado inicial de G, e $\overline{\mu} = [I - \Psi]^{-1}\overline{\chi}$, em que $\overline{\chi} \equiv [\chi_1 \, \chi_2 \ldots \chi_n]^T$ é um vetor com os pesos da função característica para cada estado de G.

A medida de linguagem para um supervisor implementado pelo autômato $Z = (Q_2, \Sigma, \delta_2, q_{2i}, Q_{2m})$ para a planta $G = (Q_1, \Sigma, \delta_1, q_{1i}, Q_{1m})$ é $\mu(L_m(G) \cap L_m(Z))$, em que o

custo de evento é obtido por $\widetilde{\psi}[\sigma,(q_{1i},q_{2j})] = \widetilde{\psi_1}[\sigma,q_{1i}]$ e a função característica é $\chi((q_{1i},q_{2j})) = \chi_1(q_{1i})\Upsilon(q_{2j})$, em que $\Upsilon(q) = 1$ para $q \in Q_{2m}$ e $\Upsilon(q) = 0$ para $q \notin Q_{2m}$ (SURANA e RAY, 2004).

Um passo importante a ser notado para que se possa comparar a medida de linguagem de supervisores $SupC(K_{aj}, G_{aj})$ baseados em diferentes planta aproximadas G_{aj} é estabelecer uma planta comum para obtenção do custo de evento. Este trabalho sugere que G_a, com $L(G_a) = \Pi^{-1}(L(G))$ e $L_m(G_a) = \Pi^{-1}(L_m(G))$, seja esta planta base. A razão para esta sugestão é que G_a é a aproximação mais simples para G_d e que $L_m(G_{aj}) = L_m(G_a) \cap L_D$ e $SupC(K_{aj}, G_{aj}) \subset L_m(G_a)$ para todas as plantas aproximadas G_{aj}, também para G_d. Além disso, pode ser mostrado que $\mu(SupC(K_{aj}, G_{aj}) \cap L_D \cap L_m(G_a)) = \mu(SupC(K_{aj}, G_{aj}) \cap L_D)$ quando considera-se G_a como planta base para obtenção do custo de evento. Por fim, G_a possui potencialmente menos estados que qualquer outra planta aproximada G_{aj}, tornando a medida de linguagem mais fácil de ser computada.

A medida de linguagem pode ser intuitivamente relacionada a quão permissivo um supervisor é. Dado que a medida leva em consideração o número de cadeias terminando em um estado marcado, quanto mais cadeias terminando em estados marcados, maior a medida será, ou, em outras palavras, menos restritivo será o supervisor. Adicionalmente, pode ser demonstrado que, para o conjunto de valores determinado para aplicação neste trabalho, a medida de linguagem é um número real não-negativo. Desde que as linguagens $SupC(K_{aj}, G_{aj})$, para todas as plantas aproximadas G_{aj}, estão contidas na linguagem marcada da planta aproximada mais simples G_a, pode ser demonstrado que os valores para $\mu(SupC(K_{aj}, G_{aj}) \cap L_D)$ são limitados por $\mu(L_m(G_a))$, podendo este valor ser usado como fator de normalização das medidas.

3.2 Busca heurística

Na seção anterior foi exposto como a aproximação de distinguidores pode ser usada para reduzir a complexidade da síntese de um supervisor. Devido às razões mencionadas na introdução desta seção, este trabalho propõe o uso de heurística para explorar o espaço formado por supervisores obtidos por distinguidores aproximados. O algoritmo genético (PARDALOS et al., 2011; BROWNLEE, 2011) foi então escolhido para ser a heurística embutida na busca.

O algoritmo genético é baseado na evolução biológica, em que indivíduos da geração denominada *pais* se combinam para criar uma nova geração denominada *filhos*. Neste processo, o número de descendentes de um indivíduo é proporcional à sua adaptação ao meio, de forma que é mais provável que seu material genético esteja presente nas futuras gerações. Alguns erros aleatórios na transcrição do material genético pode ocorrer durante a reprodução, causando mutações. Estas mutações podem degradar a adaptação ao meio, assim como pode aprimorá-la (BROWNLEE, 2011).

O algoritmo genético aplicado a problemas de otimização envolvendo buscas visa a encontrar o melhor candidato à solução. A analogia inclui ainda a *função custo*, que mede a adaptação do candidato, e a representação do candidato em código genético. Esta representação deve permitir que o algoritmo gere o candidato para que a função custo possa medir algum requisito que o algoritmo pretende maximizar ou minimizar.

Para o presente trabalho, os candidatos são supervisores obtidos por distinguidores aproximados, sendo representados pelo código genético em que cada gene indica se um módulo específico, dentre aqueles que compõem o distinguidor preditivo, está presente ou não na aproximação do distinguidor usado na síntese. Por exemplo, uma aproximação empregando somente os módulos 1 e 3, de cinco módulos, é representado pelo gene *10100*. A função custo é a medida de linguagem computada conforme exposto na subseção 3.1.

A Busca Heurística é implementada pelo algoritmo 1, em que a função *Seleciona_Reproduz()* é responsável por selecionar os indivíduos de *pop*, proporcionalmente à sua adaptabilidade, e combinar seus códigos genéticos para gerar *filhos*.

Algoritmo 1 - Busca heurística

```
Inicializa (n_pop, pop, sup_ótimo, n_iter);
Índice(pop);
para i de 1 a n_iter faça
    para j de 1 a n_pop faça
        filhos(j) = Seleciona_Reproduz(pop);
        se NaoConflito(filhos(j), L_D) então
            se Compara(filhos(j), supΣ_u(filhos(j))) então
                sup_ótimo := filhos(j);
                parar busca;
            fim
            se Índice(filhos(j)) > Índice (sup_ótimo) então
                sup_ótimo := filhos(j);
            fim
        fim
    fim
    pop := filhos;
fim
retorna sup_ótimo;
```

A cada indivíduo criado corresponde um supervisor que é testado quanto ao conflito, função *NaoConflito()*, e, caso não seja conflitante, sua otimidade também é testada, função *Compara()*. Se o supervisor for o maximamente permissivo, a busca encerra retornando este supervisor, caso contrário, a adaptabilidade do indivíduo é obtida, função *Índice()*, sendo 0 para os que representam supervisores conflitantes. Por fim, se a adptabilidade do indivíduo for maior que a de *sup_ótimo*, o algoritmo o substitui pelo indivíduo atual. Depois de formados todos os indíviduos de *filhos*, eles passam a ser a nova *pop* e uma nova geração é criada.

Os passos da busca são a síntese de um supervisor por distinguidor aproximado, o teste de conflito e o cômputo da medida de linguagem, com ordens de complexidade de $O(ng^2 \cdot ne^2 \cdot nha^2)$, $O(ng \cdot ne \cdot nha \cdot nh)$ e $O(ng \cdot ne \cdot nh \cdot log(ng \cdot ne \cdot nh))$, respectivamente, em que nh é o número de estados do distinguidor preditivo, e ng, ne e nha são os mesmos definidos na seção 2.4. Dado que o o passo de maior complexidade é o teste de conflito, desde que nha seja significativamente menor que nh, a ordem de complexidade da busca é menor que a da síntese do supervisor por distinguidor preditivo. Deve-se ressaltar que o principal objetivo do método proposto não é desenvolver um método mais rápido para a síntese de supervisores, mas sim, criar um método para se obter o melhor supervisor possível nos casos em que a síntese com distinguidor preditivo esgota todos os recursos de uma máquina, tornando impossível a obtenção do supervisor.

4 Estudo de caso

Para ilustrar o método proposto, ele é aplicado a um sistema de manufatura, Fig. 1, com retrabalho cujo objetivo é empilhar peças oriundas de máquinas diferentes, gerando uma nova peça.

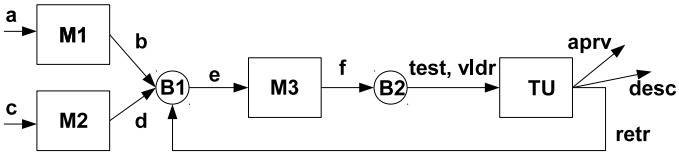
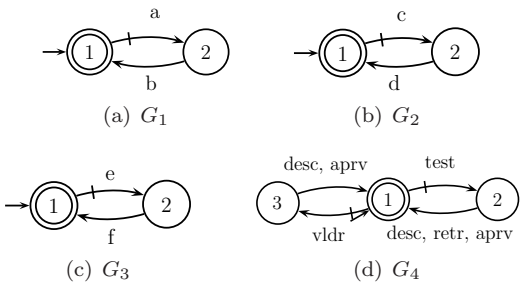


Figura 1: Linha de manufatura de empilhamento com retrabalho.

O sistema consistem em três máquinas, M_1, M_2 e M_3, uma unidade de testes, TU, e dois *buffers*, B_1, e B_2, interligando as máquinas. As máquinas M_1 e M_2 trabalham concorrentemente, depositando peças do tipo A e B, respectivamente, em B_1. A máquina M_3 coleta duas peças por vez de B_1 e as empilha, criando uma nova peça. Há quatro tipo de peças, de acordo com a origem e ordem das partes, M_1 ou M_2: AA, AB, BA, e BB. A peça é então testada em TU, que determina o seu descarte, aprovação ou retrabalho em M_3.

A Fig. 2 mostra os autômatos G_1, G_2, G_3, e G_4 para os subsistemas M_1, M_2, M_3 e TU, respectivamente. O início de operação das máquinas é modelado pelos eventos *a*, *c*, e *e*, respectivamente para M_1, M_2 e M_3, e o término de operação pelos eventos *b*, *d*, e *f*. Para o autômato G_4, o evento *test* modela o teste cujo resultado é o descarte, evento *desc*, aprovação, evento *aprv*, ou retrabalho, evento *retr*, enquanto *vldr* modela o teste cujo resultado é descarte, *desc*, ou aprovação, *aprv*.



(a) G_1 (b) G_2 (c) G_3 (d) G_4

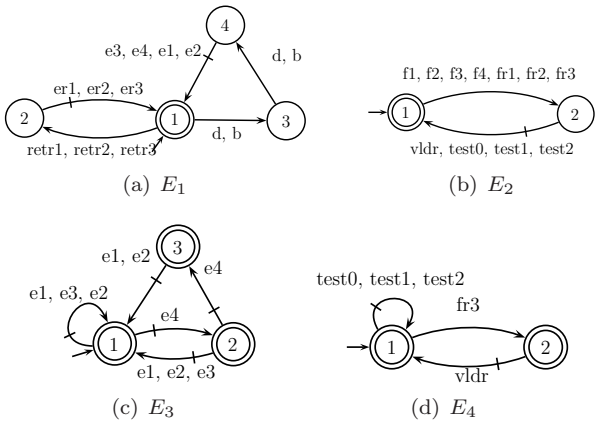
Figura 2: Modelos para os subsistemas M_1, M_2, M_3 e TU.

A planta para o comportamento do sistema em malha aberta é obtido pela composição síncrona dos quatro autômatos, $G_1 \| G_2 \| G_3 \| G_4$, em que o alfabeto, Σ, é particionado em $\Sigma_c = \{a, c, e, test, vldr\}$ e $\Sigma_u = \{b, d, f, desc, aprv, retr\}$.

Há quatro especificações para o sistema:

- $E_i, i \in \{1, 2\}$: evitar excesso de capacidade de B_i;
- E_3: estabelecer o limite de três ciclos de retrabalho para uma mesma peça;
- E_4: determinar que após uma sequência de duas peças do tipo BB, deve ser manufaturada uma do tipo AA ou AB.

A modelagem das especificações E_3 e E_4 é uma tarefa complexa, uma vez que é necessário rastrear as peças por todo o sistema para que se determine a sequência de peças produzidas e o número de retrabalhos para



(a) E_1 (b) E_2 (c) E_3 (d) E_4

Figura 3: Modelos para as especificações no alfabeto refinado Δ.

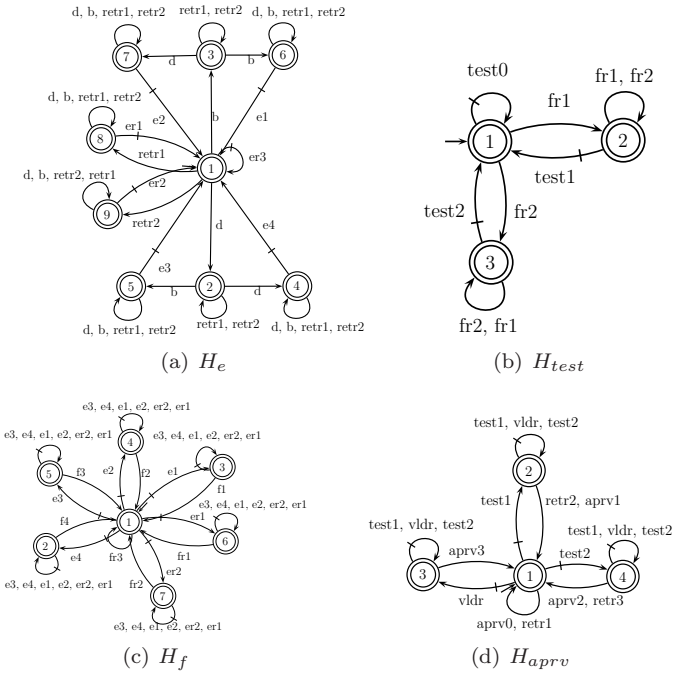
cada uma. Essa tarefa é simplificada consideravelmente pelo uso de distinguidores, introduzido como explicado a a seguir. Os eventos do alfabeto original, Σ, são refinados em suas instâncias de acordo com a peça manipulada, tal que $\Delta^e = \{e_1, e_2, e_3, e_4, e_{r1}, e_{r2}, e_{r3}\}$, $\Delta^f = \{f_1, f_2, f_3, f_4, f_{r1}, f_{r2}, f_{r3}\}$, onde o sufixo *rn* indica uma peça vinda de TU no *n_esimo* ciclo de retrabalho, $\Delta^{test} = \{test_0, test_1, test_2\}$, $\Delta^{aprv} = \{aprv_0, aprv_1, aprv_2, aprv_3\}$, em que o sufixo numérico indica o número do ciclo de retrabalho, $\Delta^{retr} = \{retr_1, retr_2, retr_3\}$, em que o sufixo numérico indica o número do ciclo de retrabalho que a peça entrará, e $\Delta^\sigma = \{\sigma\}, \sigma \in \{a, b, c, d, vldr, desc\}$. Por exemplo, a instância f_{r2} modela o fim de processamento de uma peça no segundo retrabalho na máquina M_3. O novo alfabeto refinado é $\Delta = \dot{\bigcup}_{\sigma \in \Sigma} \Delta^\sigma$. A Fig. 3 apresenta as quatro especificações no alfabeto refinado Δ.

O distinguidor foi construído modularmente pelos modelos expostos na Fig. 4 e Δ^*. O autômato H_e relaciona cada instância de e ao tipo de peças coletadas em B_1, por exemplo, se a primeira peça depositada em B_1 é de M_1 e a segunda é de M_2, a ocorrência correspondente de e é e_2. A instância padrão, eleita caso nenhuma sequência descrita se aplique, é e_{r3}. O autômato H_f relaciona as instâncias de f com as correspondentes de e. O autômato H_{test} relaciona as instâncias de *test* com o número de ciclos de retrabalhos já executados para a peça a ser testada, enquanto H_{aprv} relaciona as instâncias de *aprv* e *retr* com as correspondentes de *test*.

A ordem de complexidade da síntese do supervisor por distinguidor preditivo é estimado em $O(611712^2)$, enquanto para o supervisor com a pior aproximação a ordem é de $O(864^2)$. Análises prévias indicaram que supervisores obtidos de aproximações que não contenham o módulo H_e são conflitantes com a linguagem do distinguidor preditivo e o supervisor obtido com a aproximação que contém somente este módulo, apesar de não-conflitante, é sub-ótimo.

Com a finalidade de aumentar o número de candidatos na busca, o distinguidor foi modularizado em mais um nível, gerando 21 módulos, cada um distinguindo apenas uma instância das demais da mesma máscara. Por exemplo, o módulo H_{f1}, ilustrado na Fig. 5, distingue somente a instância f_1 das demais da mesma máscara, f.

O método de busca proposto foi aplicado ao sistema em 10 testes não correlatos, todos com a população de 22 indivíduos, taxa de mutação de 5% e número de geração limitado a 40, que representa um máximo de 0,042% do espaço de busca. A tabela 1 mostra os resultados dos testes, em que a coluna *Teste* indica o número do teste, *Zo*



(a) H_e (b) H_{test}

(c) H_f (d) H_{aprv}

Figura 4: Modelos para os módulos distinguidores.

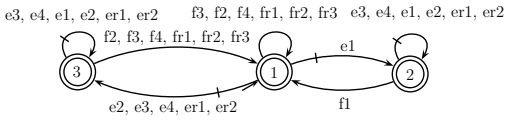


Figura 5: Distinguidor de módulo por instância.

indica quantos supervisores foram testados, H indica os módulos usados na aproximação da solução encontrada e *resultado* indica se o supervisor retornado é ótimo, conflitante com a linguagem do distinguidor preditivo, ou, caso seja sub-ótimo, sua medida de linguagem.

Tabela 1: Resultados para o algoritmo de busca.

Teste	Zo	H	resultado
1	571	$h_{e1}, h_{e2}, h_{er2}, h_{f1}, h_{fr2}, h_{fr3}, h_{aprv0}, h_{aprv2}, h_{aprv3}$	ótimo
2	410	$h_{e1}, h_{e2}, h_{fr2}, h_{fr3}, h_{test3}$	ótimo
3	764	$h_{e1}, h_{e2}, h_{e3}, h_{f1}, h_{fr2}, h_{fr3}, h_{aprv0}, h_{aprv2}, h_{aprv3}, h_{test2}, h_{test3}$	ótimo
4	321	$h_{e1}, h_{e2}, h_{e4}, h_{er2}, h_{fr2}, h_{fr3}$	ótimo
5	551	$h_{e1}, h_{e2}, h_{e3}, h_{e4}, h_{er1}, h_{er2}, h_{fr2}, h_{fr3}, h_{aprv0}, h_{aprv3}$	ótimo
6	420	$h_{e1}, h_{e2}, h_{e3}, h_{fr2}, h_{fr3}, h_{aprv1}, h_{aprv2}$	ótimo
7	598	$h_{e1}, h_{e2}, h_{e4}, h_{er2}, h_{er3}, h_{fr2}, h_{fr3}, h_{test1}$	ótimo
8	509	$h_{e1}, h_{e2}, h_{e3}, h_{e4}, h_{er2}, h_{er3}, h_{fr2}, h_{fr3}$	ótimo
9	645	$h_{e1}, h_{e2}, h_{er1}, h_{er2}, h_{er3}, h_{fr2}, h_{fr3}, h_{test1}$	ótimo
10	272	$h_{e1}, h_{e2}, h_{e4}, h_{er1}, h_{f1}, h_{f2}, h_{f4}, h_{fr1}, h_{fr2}, h_{fr3}, h_{test1}$	ótimo

Os módulos necessários para a síntese de um supervisor não-conflitante são *he1* e *he2*. Para um supervisor ótimo, os módulos *hfr2* e *hfr3* também são necessários. Todos os testes resultaram em supervisor ótimo, com o máximo de 764 supervisores visitados, isto é, menos de 0,036% do espaço total. Deve ser notado que, para os supervisores obtidos, a ordem de complexidade máxima é $O(321408^2)$, mas devido à construção do algoritmo, todas as aproximações anteriores à solução têm menos estados.

5 Conclusão

Este artigo mostrou como o uso do algoritmo genético pode aprimorar a busca pela melhor solução de um problema de controle supervisório em um espaço formado por supervisores sintetizados com o uso de distinguidores aproximados. Para o estudo de caso apresentado, o algoritmo alcançou a solução ótima explorando menos de 0,036% do espaço total, com uma complexidade de síntese cerca de 75% menor que a da síntese com distinguidor preditivo.

Estudos futuros podem ser realizados com a finalidade de melhorar o desempenho do método por meio de um melhor dimensionamento dos parâmetros da busca. Testes com diferentes heurísticas devem ser realizados para se determinar a que melhor se adequa à presente finalidade. Também devem ser realizados estudos sobre métodos alternativos para comparação entre as soluções retornadas, tanto para reduzir o custo computacional da operação quanto para encontrar medidas que forneçam maior distinção.

Referências

BOUZON, G., QUEIROZ, M. e CURY, J. (2008). Supervisory control of des with distinguishing sensors, *International Worshop on Discret Event Systems* pp. 22–27.

BROWNLEE, J. (2011). *Clever Algorithms: Nature-Inspired Programming Recipes*, first edn, LuLu, USA.

CASSANDRAS, C. e LAFORTUNE, S. (2008). *Introduction to Discrete Event Systems*, second edn, LLC, USA.

CURY, J., QUEIROZ, M., BOUZON, G. e TEIXEIRA, M. (2013). Supervisory control of discrete event systems with distinguishers. Artigo submetido a *Automatica*.

PARDALOS, P., DU, D. e GRAHAM, R. (2011). *Handbook of Combinatorial Optimization*, first edn, Springer, USA.

QUEIROZ, M. e CURY, J. (2002). Synthesis and implementation of local modular supervisory control for a manufacturing cell, *Proceedings of the Sixth International Workshop on Discrete Event Systems (WODES'02)* .

RAMADGE, P. e WONHAM, W. (1989). The control of discrete event systems, *Proceedings of IEEE, Special Issue on Discrete Event Dynamic Systems* **77**(1): 81–98.

SURANA, A. e RAY, A. (2004). Measure of regular languages, *Demonstratio Mathematica* **XXXVII**(2): 485–503.

TEIXEIRA, M., CURY, J. e QUEIROZ, M. (2011). Local modular supervisory control of des with distinguishers, *IEEE International Conference on Emerging Technologies and Factory Automation* .

WANG, X. e RAY, A. (2002). Signed real measure of regular languages, *Proceedings of the American Control Conference* **5**(18): 3937–3942.

WONHAM, W. e RAMADGE, P. (1988). Modular supervisory control of discrete event systems, *Math Control Signals Systems* **1**(1): 13–30.