Programación (secuenciación) pseudo-óptima de multiples productos en una estación flexible usando aprendizaje reforzado

Carlos D. Paternina Arboleda*

Resumen

Este artículo considera una estación flexible de manufactura que atiende múltiples productos y con «buffers» (sistemas de almacenamiento) de salida para cada uno de los tipos de productos disponibles. Se asume suministro infinito de materia prima y que los tiempos de producción de la estación, asociados a los diferentes productos, son variables aleatorias. Además, el proceso estocástico de demanda, independiente para cada producto, se considera de tipo Poisson. El objetivo del problema es determinar una secuencia óptima (pseudo-óptima) de producción que maximice la ganancia neta. El problema puede ser expresado como un problema de decisión semi-markoviano (SMDP [9]) y puede ser resuelto por técnicas convencionales de programación dinámica. Sin embargo, la magnitud del espacio de estados posibles para este problema hace que la solución por medios convencionales de programación dinámica sea prohibitiva, debido a que la obtención de la matriz de probabilidades de transición entre estados es difícil de obtener. La secuencia pseudo-óptima de producción es encontrada por medio de una técnica de optimización con simulación basada en inteligencia computacional llamada «aprendizaje reforzado» (reinforcement learning *[3, 4, 6]).*

Palabras claves: Secuenciación de operaciones, aprendizaje reforzado, simulación, optimización.

Abstract

This paper considers a single flexible station with multiple products and multiple output buffers. It is assumed that there is infinite supply of incoming raw parts and that production times are random variables. Also, the demand arrival process follows an independent Poisson process for each product. The problem aims at determining a production schedule that maximizes the net profit. The problem can be set up as a semi-markov decision problem (SMDP [7]) and can be solved by Dynamic Programming (DP) techniques. However, the problem under study is too hard for DP techniques because of the huge state space and the complex SMDP underlying process, which makes the transition probabilities difficult to obtain. Hence, a simulation-based computational intelligence technique for dynamic optimization, called Reinforcement Learning [3, 4, 6] is used.

Key words: Scheduling, reinforcement learning, simulation, optimization.

Fecha de recepción: 10 de octubre del 2000

* Profesor del Departamento de Ingeniería Industrial de la Universidad del Norte. Laboratorio de Robótica y Automatización de la Producción. *(e-mail: cpaterni@uninorte.edu.co)*

1. INTRODUCCIÓN

El sistema en consideración consta de una estación flexible de manufactura, capaz de producir múltiples productos y con múltiples *buffers*. En cualquier instante se asume que la estación de producción tiene suministro infinito de material. Los tiempos de producción se asumen como variables con distribución de probabilidad general, esto es, no siguen propiedades de memoria relacionadas con procesos exponenciales. En adición, el proceso de demanda para cada producto es una variable aleatoria de tipo Poisson. Demanda no satisfecha se pierde en el sistema. La función de costos asociada a este problema otorga un beneficio por cada parte que satisface una entidad de demanda (requisición). Cada vez que se cambia de tipo de producto, el sistema incurre en un costo de alistamiento. No se consideran costos de oportunidad debido a demanda insatisfecha. Se asume que la estación no está sujeta a fallas durante el horizonte de operación del sistema (tiempo de simulación).

Cuando se termina de fabricar un producto se deposita en su respectivo *buffer*. Además, cuando una entidad de demanda llega al sistema se lleva consigo una unidad producida, siempre que haya disponibilidad. Al final de un ciclo de producción, el agente debe decidir entre una de las siguientes acciones:

- Continuar producción del mismo tipo de producto
- Cambiar a otro producto
- Mantener la estación inactiva hasta el próximo evento de demanda.

La decisión tomada por el agente inteligente se basa en los estados (niveles) de almacenamiento de los respectivos productos, los costos de alistamiento, el beneficio generado por el servicio a la demanda, los procesos de demanda y de producción. Por ejemplo, es apenas lógico pensar en cambiar de tipo de producto cuando los niveles del producto que actualmente se fabrica están altos y los de otro producto están bajos. Además, el proceso de optimización es específico. Dos procesos diferentes de demanda pueden conducir a diferentes acciones por parte del agente. Todo se basa en la interacción existente en el sistema y la función de costos que se desea optimizar.

El problema de aprendizaje puede ser resumido de la siguiente forma. Después de que el agente toma una decisión, el simulador la ejecuta y envía una respuesta que es alimentada a la función de aprendizaje del agente. El sistema evoluciona hasta alcanzar estabilidad computacional en los valores de la matriz de pagos[1] [7, 8]. Después de alcanzar estabilidad se ejecuta el sistema con el controlador (agente) sin función de aprendizaje (fase de prueba) y se obtienen los resultados. La figura 1 esquematiza la actividad de aprendizaje del agente inteligente:

[1] La *matriz de pagos* corresponde al conjunto de valores que el agente de aprendizaje verifica para tomar una decisión. Dicha matriz es modificada durante el proceso de aprendizaje hasta alcanzar estabilidad en la función objetivo.

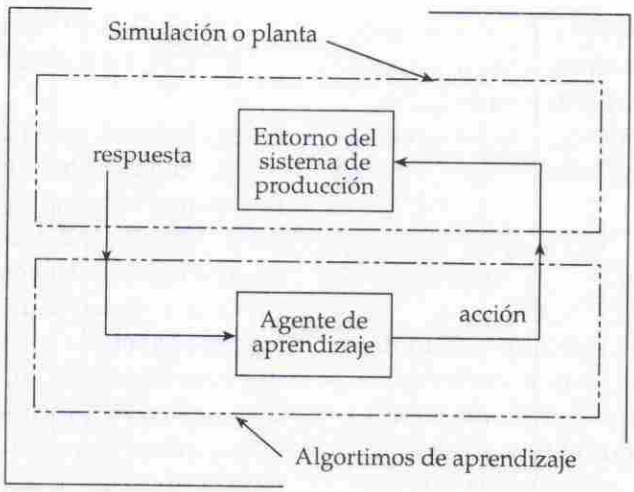

Figura 1. Esquema típico de *aprendizaje reforzado*

2. MODELO MATEMÁTICO

Primero se identifican los procesos estocásticos que relacionan los eventos aleatorios que gobiernan el sistema, en particular, los procesos de producción y de demanda. El lector debe notar que no se presentan las ecuaciones para las probabilidades de transición de un paso. El sistema está representado por un vector Q de dimensión $(n+1)$. El vector está dado por:

$$Q = \{\omega, b_1, b_2, \ldots, b_n\}, \qquad \text{(ecuación 1)}$$

donde ω es el estado de la estación flexible de producción al momento de la toma de decisiones por parte del agente, y b_1 es el nivel de los *buffers* de almacenamiento de cada tipo de producto al momento de la toma de decisiones.

El sistema cambia de estado siempre que uno de los componentes del vector se modifique. Sin embargo, el interés de estudio bajo el esquema de procesos de decisión semi-markovianos es el de enfocar el sistema hacia los estados en los cuales se debe tomar una decisión. La razón por la cual se utiliza este enfoque es para encontrar una política pseudo-óptima de secuenciación de productos en la estación flexible. Los estados en los cuales se toman decisiones se encuentran sólo cuando se completa una operación de producción o cuando llega una entidad de demanda y la máquina se encuentra inactiva.

Cuando el sistema ejecuta una acción promovida por el agente y alcanza otro estado en el cual se requiere una decisión, se dice que una *época* ha sido ejecutada. Si se define X_m como el estado del

sistema en la época m y T_m es el tiempo transcurrido para alcanzar dicha época, entonces X_m sigue cierta propiedad y se considera que el proceso está asociado a una cadena de Markov:

$$P\{X_{m+1}=j \mid X_0, \ldots, X_m; T_0, \ldots, T_m\} = P\{X_{m+1} = j \mid X_m; T_{m-1} - T_m\}, \quad \text{(ecuación 2)}$$

Además, si Y_t indica el estado del sistema en el instante t, entonces es obvio que Y_t es un proceso semi-markoviano, debido a que el tiempo transcurrido entre las épocas m-ésima y $(m+1)$-ésima es una variable aleatoria [9].

Con el objetivo de obtener variables que midan el desempeño del agente inteligente (controlador), se define a continuación una estructura de costos (beneficios). Si el sistema es corrido por un tiempo de simulación τ bajo una política fija de secuenciación y si Cp_i es la ganancia obtenida al satisfacer una entidad de demanda para el producto i-ésimo y $Cs_{i,j}$ es el costo de alistamiento de la estación cuando pasa de fabricar del producto i al producto j, entonces la función de ganancia promedio se define como (sin incluir costo de almacenamiento)

$$\rho = \frac{1}{\tau}\left(\sum_i Cp_i d_i - \sum_i \sum_j Cs_{i,j} n_{i,j} \right), \text{(ecuación 3)}$$

donde d_i representa el número de entidades de demanda para el producto i-ésimo que fueron satisfechas en t unidades de tiempo, y $n_{i,j}$ representa el número de veces que el sistema cambió del producto i al producto j en τ unidades de tiempo.

3. APRENDIZAJE REFORZADO *(REINFORCEMENT LEARNING)*

Las métodos de aprendizaje reforzado son procedimientos numéricos iterativos (sincrónicos o asincrónicos) de aprendizaje de máquina para procesos de decisión estocásticos, en los cuales se premian o penalizan las acciones del agente de acuerdo con el resultado producto de dichas acciones. El agente inteligente no es informado sobre las acciones que se van a tomar previamente, para permitir que el aprendizaje se produzca por medios de exploración y explotación del espacio de soluciones.

Los procedimientos de aprendizaje de máquina permiten enseñar a los agentes decisores cómo predecir una política de control. Esto se logra mediante la asignación de costos y ganancias a una función de aprendizaje, de acuerdo con las acciones que dicho agente tome a lo largo de su aprendizaje. El agente inteligente proporciona al sistema el conjunto de acciones, y en respuesta obtiene las entradas que determinan la siguiente decisión que se debe tomar. El aprendizaje reforzado no es un tema nuevo en la literatura de Inteligencia Artificial. Se relaciona con trabajos en áreas tan diversas como estadística, neurociencias, ciencias de la computación e investigación de operaciones. Se caracteriza por ser una alternativa factible para la solución de procesos de decisión de Markov (MDPs).

En la comunidad académica computacional, el término «reinforcement»

(palabra inglesa que traduce «refuerzo») es visto como una *clase de problemas* en lugar de un conjunto de técnicas (más comúnmente utilizado por la comunidad de psicología). A continuación se presenta el algoritmo SMART (técnica de ganancia promedio para procesos semi-markovianos) presentado por Das y colaboradores [3] y Gosavi [4].

1. Haga el contador de época $m = 0$. Inicializar la función de valores por acción $R_{old}(i, a) = R_{new}(i, a) = 0, \forall \iota \in E, \wedge a \varepsilon A_i$. Escoja el estado inicial i arbitrariamente. Haga la función de ganancia acumulada $c_m = 0, t_m = 0, \rho_m = 0, \forall a \in A_i, \forall i \in E$.

2. *Haga por siempre (hasta encontrar convergencia)*

(a) Desarrolle la acción a. El estado en la siguiente época de decisión es j, $\tau(i, j, a)$ es el tiempo de transición debido a la acción a, y $r_{imm}(i, j, a)$ es la ganancia inmediata resultado de tomar la acción a en el estado i.

(b) Encuentre $R_{m+1}(i, a)$, como sigue:

- $R_{m+1}(i, a) \leftarrow (1 - \alpha_m) R_m(i, a) + \alpha_m\{g(i, j, a) - \rho_m \tau(i, j, a) + \max_b R_m(j, b)\}$

(c) En caso de que se escoja una acción no aleatoria en el paso 2(a)

- Actualice el tiempo total: $t_m \leftarrow t_m + \tau(i, j, a)$
- Actualice la ganancia total para el agente: $c_m \leftarrow c_m + g(i, j, a)$
- Actualice la ganancia promedio: $\rho_m \leftarrow c_m/t_m$

(d) Haga el estado actual i el nuevo estado j, y $m \leftarrow m + 1$. Actualice tasas de exploración y de aprendizaje.

FIN del lazo

Algunas consideraciones adicionales al presente algoritmo se presentan en Gosavi [4]. SMART diverge para algunas condiciones de costo restrictivas y diversas. Para aliviar este problema, el mismo autor presenta la adición del algoritmo estocástico de Robins-Monroe, cuyo objetivo principal es el de garantizar convergencia del algoritmo. Este nuevo algoritmo se denomina *Relaxed*-SMART. La característica principal se define a continuación:

$$\rho^{m+1} = (1 - \beta(m))\rho^m + \beta(m) + \frac{\lfloor \tau(m)\rho^m + g(i,u,e_{iu}^m) \rfloor}{\tau(m+1)} \qquad \text{(ecuación 4)}$$

esto es, la actualización de la función de ganancia se determina de manera estocástica para garantizar su convergencia. El algoritmo *Relaxed*-SMART no sólo garantiza convergencia, sino que además garantiza que el valor de la función de ganancia presente convergencia al valor óptimo.

4. EJEMPLO NUMÉRICO

Considere un sistema de producción en el cual se tiene una estación flexible de manufactura con capacidad para atender 3 productos. El sistema tiene las siguientes características de operación durante un período de 240.000 unidades

de tiempo con tiempo de calentamiento de 9.600 unidades:

- Tiempos de producción distribuidos Gamma para cada producto con parámetros $(n, \lambda) = (3, 1/8)$,
- Proceso de demanda Poisson con tasa 1/60 para cada producto,
- Las ganancias por satisfacción de unidades de demanda son 10, 12 y 15 para los productos 1, 2 y 3, respectivamente,
- El costo de alistamiento de la estación es de 2, 1.5 y 2 para los productos 1, 2 y 3, respectivamente (sin importar el producto de la operación previa).

Resultados de la optimización

Función de ganancia	Niveles de Servicio		
	v^1	v^2	v^3
58.86	0.91	0.91	0.94

Note que los niveles de satisfacción de la demanda son superiores en todo caso al 90%. Si la función de optimización que se desee utilizar restringiera el problema a casos en los cuales se cumpla un nivel de servicio superior a un valor β predeterminado, el resultado de la optimización sería diferente. El lector puede remitirse a Paternina y Das [8] para una discusión más profunda al respecto.

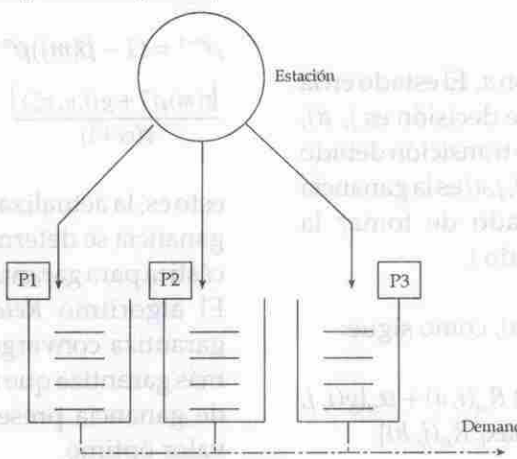

Figura 2. Estación flexible de manufactura con tres productos

El espacio de estados explorados es de aproximadamente 15.000 estados. El sistema evoluciona y es capaz de aprender la política que maximiza la función de ganancia propuesta (ecuación 3). La tabla muestra los resultados del problema de optimización.

La figura 3 muestra el comportamiento del agente inteligente durante su fase de aprendizaje. Note que al inicio de la simulación el nivel de exploración del agente es mayor y a medida que la simulación va transcurriendo el agente disminuye su patrón de exploración y

la función de aprendizaje converge al óptimo. Para mayor explicación sobre el comportamiento del algoritmo *Relaxed*-SMART y la demostración matemática de su convergencia, el lector debe remitirse a Gosavi [4].

La arquitectura de optimización propuesta es genérica y puede ser aplicada a diversos sistemas productivos en los cuales se involucren políticas dinámicas de control. Sin embargo, cada vez que un sistema sea modificado, el agente debe ser actualizado mediante el aprendizaje por simulación.

Figura 3. Convergencia computacional del agente para la función de ganancia promedio

CONCLUSIONES

La adaptación de técnicas de inteligencia artificial para optimización a las metodologías de simulación facilita en gran medida el trabajo de análisis de sistemas de producción para el mejoramiento de los parámetros de operación y, por consiguiente, de la productividad.

La flexibilidad que agrega el análisis de optimización «off-line» a los sistemas de producción permite la implementación de agentes inteligentes capaces de responder a las exigencias de entornos dinámicos, tales como procesos de demanda, logística y producción.

Aun para procedimientos de aprendizaje reforzado, problemas con mayor complejidad en su estructura de pagos y en el número de estados posibles se vuelven prohibitivos. Para solucionar este inconveniente se proponen medios de aproximación de la función que representa la matriz de pagos, tales como esquemas de regresión incrementales como los filtros adaptativos (redes neuronales [5]). Esto puede ser observado con mayor detalle en [3, 7 y 8].

El desarrollo del proceso de optimización para una función que involucre costos de almacenamiento y de órdenes represadas *(backlogs)* será objeto de estudios posteriores.

Referencias

1. R. ANUPINDI and S. TAYUR. «Managing Stochastic Multiproduct Systems: Model, Measures, and Analysis». *Operations Research*, 46, 3, S98-S111, 1998.
2. C. DARKEN, and J. E. MOODY. «Towards Fasters Stochastic Gradient Search». In *Advances in Neural Information Processing Systems 4* (J.E. Moody, S. J. Hanson and R. P. Lippmann (eds.). San Mateo, CA: Morgan Kaufmann, 1992.
3. T. K. DAS, A. GOSAVI, S. MAHADEVAN and N. MARCHELLACK. Solving Semi-Markov. «Decision Problems using Average Reward Reinforcement Learning». *Management Science*.
4. A. GOSAVI. «An Algorithm for Solving Semi-Markov Decision Problems Using Reinforcement Learning: Convergence Analysis and Numerical Results». Unpublished Ph.D. Thesis, University of South Florida, Department of Industrial Engineering, 1999.
5. S. HAYKIN. *Neural Networks: A Comprehensive Foundation*. Englewoods Cliffs, McMillan College Publishing Company, 1994.
6. L. P. KAELBLING, M. L. LITTMAN and A. W. MOORE. «Reinforcement Learning: A Survey». *Journal of Artificial Intelligence Research*, 4, 237-285, 1996.
7. S. MAHADEVAN and G. THEOCHAURUS. «Optimizing Production Manufacturing using Reinforcement Learning». *Eleventh International FLAIRS Conference*, p. 372-377. AAAI Press, May 1998.
8. C. D. PATERNINA-ARBOLEDA and T. K. DAS. «Intelligent Dynamic Control of Single-Product Serial Production Lines». To appear in *IIE Transactions*, Special Issue on Design and Manufacturing, 2000.
9. M. L. PUTTERMAN. *Markov Decision Processes*. Nueva York, Wiley Interscience, 1994.