Inteligencia Artificial

Estado del Arte: Car Sequencing Problem

Daniel Alejandro Martínez Castro

September 30, 2022

Evaluación

Resumen (5%):	_____
Introducción (5%):	_____
Definición del Problema (10%):	_____
Estado del Arte (35%):	_____
Modelo Matemático (20%):	_____
Conclusiones (20%):	_____
Bibliografía (5%):	_____
Nota Final (100%):	_____

Abstract

Car Sequencing Problem (CS) es un problema en el que se ordenan autos en una línea de producción, con el fin de instalar opciones, estas se instalan en estaciones de trabajo las cuales tienen una capacidad definida, cada auto pertenece a una clase la cual tiene un set de opciones predefinido y una demanda. El objetivo de este problema para una instancia es encontrar una secuencia que sea factible o declarar la instancia como sin solución, aunque, existen enfoques que permiten que se presente como solución la que cometa menos errores, para esto se utilizan restricciones blandas. En este escrito se explica el problema en detalle para luego mostrar las diferentes aproximaciones que se han realizado a este problema con el pasar del tiempo y sus variantes, junto con las heurísticas utilizadas para resolverlo, finalmente se expone un modelo matemático y conclusiones.

1 Introducción

Car sequencing problem (CS) es un problema en el que se busca ordenar una cantidad de autos de diferentes clases en una línea de producción para satisfacer la demanda, cada auto tiene asociada una clase la cual tiene definida un set opciones que se le deben instalar en diferentes estaciones de trabajo, estas estaciones tienen una capacidad asociada.

El CS es considerado un CSP donde las soluciones son del tipo SI/NO. SI en el caso que se ha encontrado una solución posible que satisface las capacidades de las estaciones y No en el caso contrario, aunque también existen modelos que permiten que una restricción pueda ser violada en este caso se usa como solución la que tenga la menor cantidad de violaciones.

Este problema es de particular interés para la industria automotriz por sus amplias aplicaciones prácticas, debido a que la industria se encuentra en crecimiento constante y en esta ocurren cambios mayores cada muy poco tiempo, como lo son los cambios tecnológicos, además de que se manejan procesos muy complejos por lo que se hace necesario tener un método para poder

organizar la producción y con esto satisfacer la demanda cambiante de la actualidad.

En este documento se encuentra estructurado siguiente manera, en la sección 2 se define el problema, explicando sus constantes, variables y restricciones. En la sección 3 se detalla el estado del arte en el cual se resume investigaciones realizada por múltiples investigadores, y sus enfoques en torno a resolver el problema. En la sección 4 se presenta el modelo matemático general en el cual se formalizan todos los aspectos del problema para finalmente presentar conclusiones en la sección 5.

2 Definición del Problema

CS un problema en que se busca ordenar autos que tienen asociada una clase en una linea de producción de autos, en la cual se le deben instalar un set de opciones a cada clase de autos, para instalar estas opciones se tiene estaciones de trabajo, una por cada opción, las que tienen una capacidad definida. El objetivo de este problema es minimizar la cantidad de violaciones de capacidad de las estaciones satisfaciendo la demanda de cada una de las clases de autos [3].

Para la resolución de CS normalmente se usan 3 restricciones:

1. Satisfacer la demanda de todas las clases.
2. No sobrepasar la capacidad de una estación de trabajo en ninguna ventana de tiempo.
3. Asociar los autos que una clase en un slot definido con las opciones que se instalan en él.

La primera y tercera restricción son duras por lo que se deben cumplir en todas las instancias del problema, en cambio la segunda restricción es blanda por lo que puede ser violada, pero manteniendo estas violaciones en lo más mínimo posible.

Las constantes para la resolución de este problema consisten en el número de opciones, número de clases, cantidad de autos totales a fabricar, demanda de cada una de las clases, set de opciones que se le deben instalar a cada clase y capacidad de las estaciones de trabajo la cual se representa de manera porcentual por otro lado las variables de este problema consisten en una lista S en la cual cada slot representa una clase de auto que se fabricara en ese intervalo de tiempo, y una matriz O en la cual las columnas representan las opciones que se le instalan a un auto en un intervalo de tiempo.

Este problema se ha probado ser NP-duro por Kis [8], lo que significa que puede que no se pueda resolver con algoritmos en tiempo polinomial. El espacio de búsqueda corresponde a todas las posibles permutaciones de diferentes clases de vehículos, pero la dificultad no solo depende del espacio de búsqueda, sino también de la tasa de uso de cada una clase de auto, debido a mientras más se use una clase de auto más difícil es que se cumpla con la restricción de capacidad de las estaciones de trabajo [9].

Existen múltiples variantes del CS, las cuales expanden el caso base usando demanda parcial incierta para resolver el problema, añadiendo así nuevas clases de autos con diferentes opciones a las ya definidas las cuales requieren estaciones de trabajo especializadas, el objetivo de este problema además de minimizar la cantidad de errores es también minimizar los costos que implica fabricar estos nuevos autos [1]. Otras variaciones de este problema implican añadir líneas de producción con lo cual además de los objetivos base del problema se añade el de balancear la carga en estas líneas.

3 Estado del Arte

El problema Car sequencing Problem (CS) fue definido por primera vez en 1986 por Parello et al. [10]. Este problema fue definido de la misma forma que se presenta en secciones anteriores a esta, las soluciones al problemna en este modelo consisten en encontrar un orden en los autos, en donde, todas las restricciones sean satisfechas o declarar la instancia como sin solución.

Como se expuso anteriormente este problema es NP-duro esto fue propuesto por Gent [5] y luego comprobado por Kis [8], por lo tanto esto significa que no exite un algoritmo que pueda solucionar el problema en tiempo polinomial.

En 1994 Hindi y Ploszaski [7] desarrollaron un algoritmo greedy, es estos algoritmos se inicia con una secuencia vacía y luego se le añaden autos iterativamente en relación con una función greedy, el algoritmo cual probó ser efectivo encontrando soluciones óptimas en tiempo bajos, además podía detectar que la instancia no tenía solución también en tiempos bajos en la mayoría de los casos. Pero para instancias grandes se concluyó que era necesaria la inclusión de algoritmos heurísticos auxiliares. Este trabajo cimentó las bases para las investigaciones futuras sobre el tema.

En 1995 Warwick et al.[13] propuso un Genetic Algorithm (GA) para resolver este problema. Estos consisten en técnicas que exploran el espacio de búsqueda mediante una simulación de evolución, esto se logra combinando estructuras de datos que obtuvieron buen rendimiento minimizando o maximizando la función objetivo, las soluciones de algoritmos GA convergen a valores cercanos a soluciones óptimas.

La mayoría de los trabajos que resuelven el CS usando Constraint Programing consideran el resultado final una solución o indican que la instancia no tiene solución ya que todas las restricciones se deben cumplir para que una solución candidata se considere solución final, por lo tanto, es un modelo exacto. En 2001 Bergen et al. [2] propuso un modelo que incluye tanto restricciones duras como restricciones blandas, el cual funcionaba bien con instancias pequeñas, pero se quedaba atrás con instancias más grandes o complejas. Desde este punto de vista después se desarrollaron algoritmos usando heurísticas de ordenamiento con el fin de reducir el espacio de búsqueda, estos acercamientos ponen las clases de autos más difícil o con más demanda al principio de la secuencia para así trabajar en base a ellos.

Solnon en 2001 [12] propuso un algoritmo Ant Colony Optimization (ACO) para CSP. En el cual se utiliza el comportamiento de las hormigas con el fin de encontrar una solución, estas hormigas forman una solución dejando un rastro de feromonas que luego iterativamente las hormigas posteriores siguen para mejorar la solución, estas feromonas eventualmente se evaporan. Este algoritmo se ha mejorado con el tiempo incluyendo técnicas de Local Search y Greedy Heuristics [9].

Múltiples algoritmos de Local Search (LS) se han planteado algunos más centrados en CS que otros, esto algoritmos inician con una solución candidata y luego iterativamente se mueven hacia una solución vecina de la inicial, solución vecina es un set de soluciones potenciales que varían de la solución actual por lo mínimo posible un ejemplo de esto es Puchta y Gottlieb [11] en 2002. En 2003 Puchta y Gottlieb [6] compararon los rendimientos de algoritmos LS y ACO para este problema, en estas pruebas los algoritmos que usan ACO se mostraron levemente superiores a los que usan LS para instancias grandes del problema, pero los autores recalcan que con mejoras LS puede mejorar. En 2008 Fliedner and Boysen [4] desarrollaron un algoritmo Branch and Bound exacto, este explota los aspectos estructurales del CS para así reducir la complejidad del problema.

Las tendencias actuales sobre la investigación del CS más que continuar desarrollando técnicas más eficientes son de expandir el modelo para poder así usarlo en casos más específicos de CS, como lo son añadir múltiples líneas de producción o incluir demanda parcial o incierta en el modelo para así asimilarlo más a la realidad. También se están haciendo estudios prácticos que miden la utilidad de múltiples algoritmos que usa mezclas de heurísticas para así compararlos en estos entornos.

4 Modelo Matemático

Para resolver este problema se utiliza un modelo matemático que minimiza la cantidad de violaciones de capacidad de las estaciones de trabajo, para esto es necesario la restricción ligada a la capacidad sea blanda. Para la creación de este modelo se usó el trabajo de Dincbas [3] para el planteo de las restricciones, y el de Lin para la función objetivo [9]. Considerando las opciones o, número de autos a producir j y que existen k clases de autos.

4.1 Constantes

- $D[k]$ = demanda de autos de clase k.
- $P[i]$ = máximo número de autos que un bloque permite para cada opción i.
- $Q[i]$ = tamaño del bloque para cada opción i.
- $M[i][k] = 1$, si la clase k necesita la opción i. 0, caso contrario.
- C = {1, 2, ..., k}, O = {1, 2, ..., i}, N = {1, 2, ..., j}, n = total de autos a producir.

4.2 Variables

- $S[j]$ = clase de auto k asignado al slot j.
- $O[i][j] = 1$, si el auto clase k en en slot j tiene instalada la opción i. 0, caso contrario.

4.3 Función Objetivo

$$min \sum_{i \in O} \sum_{j \in N} O[i][j] \cdot max\{ \sum_{j'=j}^{min\{j+Q[i]-1,n\}} O[i][j'] - P[i], 0\} \tag{1}$$

Esta función objetivo minimiza la cantidad de violaciones en relación con la restricción de capacidad de las estaciones de trabajo. para esto se crea una ventana de tamaño $Q[i]$ la cual se va moviendo por la línea de producción revisando que la cantidad de veces que se instala la opción i en esa ventana sea menor o igual a $P[i]$, para luego sumar todas las violaciones de cada una de las opciones.

4.4 Restriciones

$$\sum_{j'=j}^{min\{j+Q[i]-1,n\}} O[i][j'] \leq P[i], \forall j \in N, \forall i \in O \tag{2}$$

$$\sum_{i \in N} max\{-1 * (|S[i] - k| - 1), 0\} = D[k], \forall k \in C \tag{3}$$

$$M[i][S[j]] = O[i][j], \forall j \in N, \forall i \in O \tag{4}$$

La inecuación 2 corresponde a la restricción de capacidad, esta crea una ventana de largo $Q[i]$ en una de las opciones i de O, contando la cantidad de veces que se instala esta opción en esta ventana, esta cantidad tiene que ser menor o igual a $P[i]$ para así estar cumpliendo la capacidad de la máquina que instala la opción i, esta ventana se va moviendo hasta que no hay más autos, para luego revisar lo explicado anteriormente con las siguientes opciones. La ecuación 3 corresponde a la restricción que asegura que se cumpla la demanda para cada una de las clases, esta restriccoón básicamente cuenta la cantidad de ocurrencias de k en S, $\forall k \in C$, y revisa que sea igual a $D[k]$. La ecuación 4 representa la restricción de liga las opciones de un auto que se produce en el slot $S[j]$ a la matriz $O[i][j]$. Por lo tanto si un vehículo clase k se produce en un slot j y esa clase necesita la opción $M[i][k]$, esta opción debe estar en $O[i][j]$.

5 Conclusiones

A lo largo de este escrito se estudió el Car sequencing Problem (CS), presentando una definición de este problema junto con todas las variables que influyen en la resolución de este, para después analizar la evolución de las técnicas para la resolución de este problema las cuales a un inicio tenían un enfoque general para problemas de satisfacción de restricciones, para después evolucionar a algoritmos más complejos con el uso de heurísticas que reducen el espacio de búsqueda con el fin de especializarse en este problema, para finalmente presentar el modelo matemático.

A pesar de los fuertes avances a de finales de la década de los 90 e inicios de la primera década de los años 2000, el problema ha sufrido una disminución de los avances a partir de la segunda década de los anos 2000. Los focos principales de desarrollo a futuro son el uso de greedy algorithms suplementado con heurísticas como Local Search (LS) y Ant Colony Optimization (ACO), además de la focalización del problema para instancias más especificas del problema, haciendolo así, más similar a las condiciones que se presentan en la realidad.

6 Bibliografía

References

[1] Joaquín Bautista-Valhondo, Manuel Chica, and Ignacio Moya. Car sequencing problem con flotas de vehículos especiales. presentación. 10 2016.

[2] Michael Bergen, Peter van Beek, and Tom Carchrae. Constraint-based vehicle assembly line sequencing. 2056:88–99, 06 2001.

[3] Mehmet Dincbas, Helmut Simonis, and Pascal Van Hentenryck. Solving the car sequencing problem in constraint logic programming. *Proceedings of the European Conference on Artificial Intelligence*, pages 290–295, 08 1988.

[4] Malte Fliedner and Nils Boysen. Solving the car sequencing problem via branch & bound. *European Journal of Operational Research*, 191(3):1023–1042, 2008.

[5] Ian P. Gent and Toby Walsh. Csplib: A benchmark library for constraints. pages 480–481, 1999.

[6] Jens Gottlieb, Markus Puchta, and Christine Solnon. A study of greedy, local search and ant colony optimization approaches for car sequencing problems. *LNCS*, 2611, 06 2003.

[7] Khalil S. Hindi and Grzegorz Ploszajski. Formulation and solution of a selection and sequencing problem in car manufacture. *Computers & Industrial Engineering*, 26(1):203–211, 1994.

[8] Tamás Kis. On the complexity of the car sequencing problem. *Operations Research Letters*, 32:331–335, 2004.

[9] Long Lin. A hybrid greedy algorithm for the car sequencing problem. *2010 IEEE 17Th International Conference on Industrial Engineering and Engineering Management*, pages 719–722, 2010.

[10] Bruce D. Parrello and Waldo C. Kabat. Job-shop scheduling using automated reasoning: A case study of the car-sequencing problem. *Journal of Automated Reasoning*, 2:1–42, 1986.

[11] M. Puchta and J. Gottlieb. Solving car sequencing problems by local optimization. *Proceedings of the Applications of Evolutionary Computing on EvoWorkshops 2002*, 2279:132–142, 01 2002.

[12] Christine Solnon. Solving permutation constraint satisfaction problems with artificial ants. 2000, 04 2001.

[13] Terry Warwick and Edward Tsang. Tackling car sequencing problems using a generic genetic algorithm. *Evolutionary Computation - EC*, 3:267–298, 1995.