Optimal Scheduling Program

Contenido

Motivación

Bancolombia, una de las entidades financieras más prominentes del país, tiene un firme compromiso de mejorar significativamente la calidad de su servicio al cliente. En este empeño, la toma de decisiones basada en datos se erige como una de sus herramientas clave para alcanzar este objetivo. La analítica desempeña un papel fundamental en la obtención de una visión más precisa de las interacciones entre los clientes y el banco, convirtiéndose en el pilar fundamental de su estrategia.

Como parte esencial de su compromiso en la mejora de la experiencia del usuario, Bancolombia busca optimizar la cantidad de asesores en función de la demanda, garantizando al mismo tiempo horarios adecuados para el bienestar de su personal. Para lograr este objetivo, es imprescindible programar las jornadas laborales de los asesores de manera que se pueda ofrecer la mejor atención, asegurando la disponibilidad óptima de asesores para satisfacer la demanda de la manera más efectiva posible.

Definición del problema

Bancolombia está interesado en mejorar la programación horaria SEMANAL de los empleados de caja de 5 sucursales. Los empleados pueden estar en 4 estados:

1. **Trabaja**: El empleado esta disponible para atender clientes.
2. **Pausa Activa**: El empleado no esta disponible para atender clientes.
3. **Almuerza**: El empleado esta tomando su tiempo de almuerzo (no disponible para atender clientes)
4. **Nada**: El empleado no ha comenzado su jornada laboral o ya la termino.

El objetivo del ejercicio es definir en qué estado se encuentra cada uno de los empleados en franjas horarias de 15 minutos con el fin de minimizar la diferencia existente entre la cantidad de empleados trabajando y la demanda de empleados requerida para cada franja horaria cuando la demanda es mayor a la capacidad. Estas demandas se tienen para cada una de las franjas para todos los días de una semana de lunes a sábado.

A continuación, se brindan los detalles más relevantes del problema:

Los empleados tienen distintos tipos de contrato: (i) Tiempo Completo (TC), y (ii) Medio Tiempo (MT). De acuerdo con el tipo de contrato se tienen distintas restricciones. Por ejemplo, la jornada laboral de los empleados depende de su tipo de contrato. La tabla a continuación describe los tiempos de la jornada laboral.

Tipo de contrato	Lunes a viernes	Sábado
Tiempo Completo (TC)	7 horas diarias	5 horas
Medio Tiempo (MT)	4 horas diarias	4 horas

Nota: Los estados de Trabaja y Pausa Activa hacen parte de la jornada laboral. El tiempo de almuerzo NO constituye tiempo de jornada laboral.

Únicamente los empleados con tipo de contrato TC deben tomar el tiempo de almuerzo de forma CONTINUA los días entre semana (lunes a viernes). Este tiempo es de 1 hora y media.

La hora mínima de salida para tomar el almuerzo son las 11:30 am y la hora máxima para salir a tomar el almuerzo es a la 1:30 pm. Esto quiere decir que:

1. Una persona NO puede salir a tomar almuerzo a las 11:15 am
2. Una persona NO puede salir a tomar almuerzo a la 1:45 pm
3. Es VÁLIDO que una persona tome almuerzo de la 1:30 pm a las 3:00 pm

Nota: El sábado NO se programa tiempo de almuerzo para NINGÚN empleado.

Cualquier franja de trabajo debe durar entre 1 y 2 horas.

Los empleados con tipo de contrato MT NO se les programa almuerzo ningún día.

Todos los empleados deben comenzar su jornada laboral los días entre semana (lunes a viernes) entre las 7:30 am y 4:30 pm. Y los sábados deben iniciar su jornada laboral entre las 7:30 am y las 11:00 am.

Para los días entre semana (lunes a viernes) a los empleados de TC se les debe mantener constante: (i) la hora a la que inicio y fin de la jornada laboral, y (ii) el tiempo de almuerzo. Las Pausas Activas si pueden tomarse en diferentes horarios en distintos días.

Para los días entre semana (lunes a viernes) a los empleados de MT se les debe mantener constante la hora a la que inicio y fin de la jornada laboral. Las Pausas Activas si pueden tomarse en diferentes horarios en distintos días.

Los sábados los horarios de todos los empleados puede cambiar.

Todos los empleados deben trabajar mínimo 1 hora de forma continua para poder salir a una Pausa Activa o Almuerzo. Esto quiere decir que, si un empleado solo ha trabajado 3 franjas horarias, en la 4 franja horaria NO debe salir a Pausa Activa o Almuerzo.

Todos los empleados deben trabajar máximo 2 horas de forma continua sin salir a una pausa activa. Esto quiere decir que, si un empleado ha trabajado 8 franjas horarias, en la 9 franja horaria SI debe salir a Pausa Activa o Almuerzo.

Todos los días el horario de los empleados debe ser CONTINUO, desde que comienza la jornada laboral del empleado este solo puede estar en los estados de Trabaja, Pausa Activa y Almuerza (este último solo aplica para empleados de TC entre semana). Es decir, que el estado Nada solo puede estar activo al comienzo del día si el empleado no ha comenzado su jornada laboral o al final del día si el empleado ya completo su jornada laboral.

El último estado todos los días de la jornada laboral de los empleados debe ser Trabaja.

Debe haber por lo menos 1 empleado en el estado Trabaja en cada franja horaria que tenga una demanda mayor o igual a 1.

Datos

El archivo `Dataton 2023 Etapa 2.xlsx` tiene dos hojas demand y workers descritas a continuación:

demand

Columna	Descripción
suc_cod	Código de la sucursal
fecha_hora	Franja horaria
demanda	Demanda estimada

workers

Columna	Descripción
suc_cod	Código de la sucursal
documento	Código de identificación
contrato	Tipo de contrato

Solución al problema

A continuación, se expone la estrategia implementada para abordar el desafío planteado, teniendo en cuenta la complejidad combinatoria inherente a la asignación de horarios laborales.

Aplicación de la Programación con Restricciones

Ante la complejidad combinatoria del problema, caracterizada por la gran cantidad de posibles combinaciones de asignaciones horarias, hemos recurrido a la técnica de Programación con Restriciones (Constraint Programming (CP), CP). Este paradigma, que encuentra aplicaciones en campos tan diversos como la inteligencia artificial, las ciencias de la computación y la investigación operativa, es particularmente efectivo para problemas de esta naturaleza.

La técnica de CP permite delimitar y explorar el espacio de soluciones factibles, aplicando restricciones que reducen progresivamente las posibilidades hasta identificar las asignaciones válidas. Además, esta metodología puede complementarse con algoritmos de optimización, lo que no solo garantiza encontrar una solución, sino que también busca que esta sea la más óptima dentro del conjunto de posibles soluciones.

Metodología

Modelo de optimización

La formulación matemática del problema es la siguiente:

$$\mathbf{P}^* = \arg\min_{\mathbf{P}} \sum_{d} \sum_{f} \max(0, \mathbf{D}_{d,f} - \sum_{t} \mathbf{P}^{+}_{t,d,f})$$

$$\text{Sujeto a } C_1(\mathbf{P}) \wedge C_2(\mathbf{P}) \wedge \cdots \wedge C_n(\mathbf{P})$$

Donde:

- $\mathbf{P} \in 0,1^{N_t \times N_d \times N_f \times N_e}$: Matriz booleana que con tiene todo las posibles combinaciones de trabajadores, dias, franjas y estados.
- $\mathbf{P}^{+} \in 0,1^{N_t \times N_d \times N_f}$: Matriz booleana que con tiene todo las posibles combinaciones de trabajadores, dias, franjas y solo el estado "Trabaja".
- $\mathbf{D} \in \mathbb{Z}^{N_d \times N_f}$: Demanda en cada dìa y franja de la semana.
- d : Día de la semana.
- f : Franja del día.
- t : Trabajador.
- C_i : Es la i-jésima restricción asociada al problema.

Las restricciones $C_1, C_2, \ldots, C_n$ son las definidas dentro del problema además de otras restricciones que surgen del entendimiento de la dinámica de los horarios de trabajo y de la forma en que se planteó el problema, que puede ayudan a resolver el problema con mayor facilidad.

Técnica usada para solucionar el problema

Para la resolución de este problema se ha utilizado CP-SAT de Google, la cual es de código abierto. CP-SAT es una herramienta que combina programación por restricciones con técnicas de satisfacción de restricciones booleanas (SAT). El procedimiento inicia con la definición de las variables de decisión, las restricciones y la función objetivo. Posteriormente, CP-SAT aplica la propagación de restricciones para descartar soluciones no factibles y así reducir el espacio de búsqueda. La búsqueda heurística permite la exploración sistemática de soluciones potenciales, mejorando progresivamente su calidad. Esta herramienta busca optimizar la función objetivo y es capaz de encontrar soluciones óptimas o aproximadas, demostrando ser una solución robusta para la planificación y toma de decisiones en variados contextos.

Resultados

Los resultados se generaron en una máquina con las siguientes especificaciones de hardware: CPU Intel(R) Core(TM) i5-8250U @ 1.60GHz, dotada de 4 procesadores físicos y 8 lógicos, y equipada con 12 GB de RAM. El tiempo total de ejecución fue de 31 minutos.

Utilizando el archivo de entrada `Dataton 2023 Etapa 2.xlsx`, se logró alcanzar un puntaje de **2624**. Este puntaje representa la diferencia entre la demanda y la capacidad, en situaciones donde la demanda supera a la capacidad.

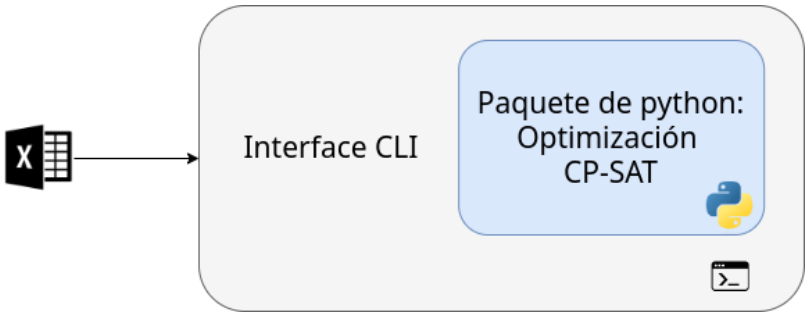
Para ver los resultados completos vaya a la siguiente carpeta ./data/resultados_etapa_2/.

Arquitectura de la Solución

La solución propuesta se estructura en torno a dos arquitecturas distintas: Arquitectura Basada en Consola y Arquitectura con Interfaz de Usuario

Arquitectura Basada en Consola

La primera arquitectura se fundamenta completamente en un paquete de Python diseñado para su uso a través de la línea de comandos. Esta elección busca maximizar la dedicación de recursos al sistema de optimización, garantizando así un rendimiento óptimo.



Arquitectura del CLI

Paquete de Python para Optimización

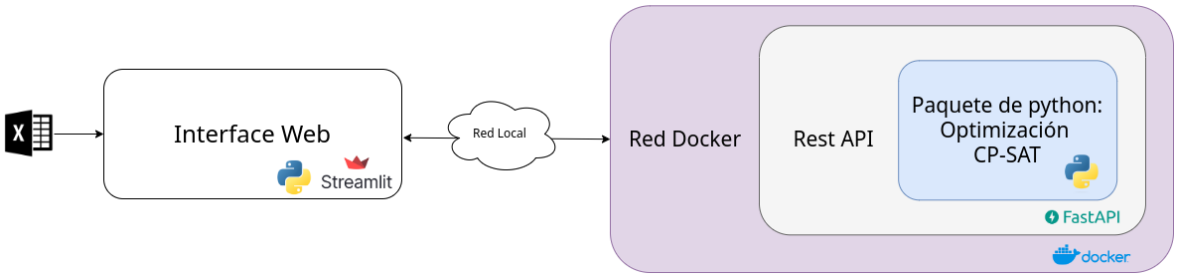
El núcleo de ambas arquitecturas es el paquete de optimización desarrollado en Python. Este paquete incorpora un modelo de optimización completo, incluyendo todas las restricciones requeridas para su funcionamiento eficaz. Además, el desarrollo de este paquete se ha llevado a cabo utilizando software de código abierto, lo que fomenta la colaboración y la transparencia.

Interfaz de Línea de Comandos (CLI)

La interfaz de línea de comandos (CLI) que hemos diseñado se enfoca en maximizar la disponibilidad de recursos para el paquete de optimización, con el objetivo de mejorar su rendimiento. A pesar de que su uso puede resultar más complejo, se ha elaborado una documentación exhaustiva que facilita su manejo y aprovechamiento. Esta interfaz es ideal para aquellos usuarios que prefieren un control detallado y directo sobre la ejecución de los procesos de optimización.

Arquitectura con Interfaz de Usuario

La segunda arquitectura está concebida para ser más accesible tanto para usuarios no técnicos como para desarrolladores. Incluye una interfaz web intuitiva y una RestAPI, facilitando la integración con otros sistemas informáticos. Esta versión se ha desarrollado a partir del paquete original de Python, utilizando FastAPI para la creación de la RestAPI y Docker para simplificar su despliegue y escalabilidad.



Arquitectura del interface we

Interfaz Web

Para facilitar el acceso de los usuarios, se ha desarrollado una interfaz web empleando Streamlit. Esta interfaz no solo simplifica la obtención de programaciones horarias, sino que también proporciona visualizaciones claras y útiles de los datos. Su diseño intuitivo y su facilidad de navegación la hacen accesible para todo tipo de usuarios.

REST API

Con el objetivo de extender la accesibilidad a desarrolladores y permitir la integración con otros sistemas, hemos implementado una REST API utilizando FastAPI. La documentación detallada de esta API, que incluye ejemplos de uso y especificaciones técnicas, está disponible a través del siguiente enlace una vez que el sistema esté en funcionamiento. Para más detalles sobre su lanzamiento, referirse a la sección correspondiente.

Despliegue con Docker

El sistema se ha contenerizado usando Docker, lo que proporciona una solución de despliegue flexible y eficiente. Docker facilita la instalación del sistema en diversas máquinas sin preocuparse por las diferencias de entorno, asegurando una configuración uniforme y una puesta en marcha rápida.

Características de la Arquitectura

La arquitectura propuesta ofrece las siguientes características clave:

- **Basado en Software de Código Abierto**: Todo el sistema se ha construido utilizando herramientas y frameworks de código abierto, garantizando transparencia, flexibilidad y una amplia comunidad de soporte.
- **Reproducibilidad con Docker**: El uso de Docker asegura la reproducibilidad y consistencia del entorno en diferentes máquinas, facilitando el despliegue y la escalabilidad del sistema.
- **Integración y Desarrollo con FastAPI**: FastAPI proporciona una base sólida para la integración con otros sistemas. Además, su diseño orientado a desarrolladores facilita la expansión y adaptación del sistema para satisfacer necesidades específicas.
- **Rendimiento Optimizado**: La arquitectura se centra en ofrecer un rendimiento sobresaliente en procesos de optimización, asegurando resultados rápidos y precisos.

Conclusiones

- Implementación de Analítica Avanzada: se ha desarrollado una herramienta para la estrategia de mejorar la calidad del servicio al cliente. Esto incluye la optimización de horarios de los asesores para coincidir con la demanda de los clientes, asegurando así una experiencia de usuario eficiente y satisfactoria.
- Solución a un Problema Combinatorio Complejo: se ha abordado la complejidad de programar horarios laborales con la técnica de Programación con Restricciones, utilizando la herramienta CP-SAT de Google. Esta aproximación permite manejar la gran cantidad de combinaciones posibles de manera efectiva, buscando la optimización y la eficiencia operativa.
- Desarrollo de una Arquitectura Flexible y Accesible: se ha desarrollado dos arquitecturas de solución distintas, una basada en la consola, para maximizar el rendimiento de la optimización, y otra con una interfaz de usuario más accesible, que incluye una interfaz web y una REST API.

Referencias

- Google Ortools - Constraint Optimization
- Using and Understanding ortools' CP-SAT: A Primer and Cheat Sheet