Memoria del Proyecto

Herramientas y Aplicaciones de la Inteligencia Artificial

David Barbas Rebollo
Aitana Sanz Rodríguez

Curso 2020-2021

Contenidos

1 Introducción

El objetivo de esta memoria es la descripción de un problema basado en una instancia real, así como su resolución mediante dos enfoques distintos y dos herramientas distintas, siendo estas Opt4J y MiniZinc. El problema a tratar es la organización de los turnos de un bar, problema posteriormente descrito y especificado, explicándose también los mecanismos de resolución obtenidos y posteriormente tanto resultados como conclusiones obtenidas.

2 Problema de optimización

La instancia del problema que va a plantearse como problema de optimización es la organización de los turnos de un bar, problema que se puede subdividir, asimismo, en dos problemas independientes, siendo estos la cocina y los camareros, puesto que no comparten todas las restricciones y la asignación de turnos de cocina no interactúa con la asignación de turnos de camareros, y viceversa. Debido a sus restricciones y tamaño, además, se puede entender el subproblema de la cocina como una instancia menos compleja del subproblema de los camareros.

2.1 Métrica: Satisfacción

La métrica a maximizar, en este caso, es la satisfacción de los empleados. Para ello, además de ofrecer una organización de turnos que sea factible, se busca ofrecer una organización de turnos que se adecue lo más posible a sus preferencias, tomadas como dato de entrada.

2.2 Subproblema: Cocina

2.2.1 Personal

El bar cuenta con un total de tres cocineros en plantilla, que se encuentran disponibles en todo momento y tienen preferencias personales respecto a los turnos que prefieren cubrir, tal y como se expone en la tabla 1. Estos cocineros, además, pueden trabajar un total de 40, 40 y 24 horas, respectivamente.

Cocinero	**Jornada**	**Disponibilidad**	**Preferencias**
Co1	40h	Siempre	Tardes
Co2	40h	Siempre	Mañanas
Co3	24h	Siempre	Tardes

Tabla 1: Cocineros en plantilla

2.2.2 Turnos

Existen un total de trece turnos de cocina, todos ellos de un empleado, siendo estos siete turnos de 09h a 17h de lunes a domingo y seis turnos de 15h a 23h, de lunes a sábado. Hay, por tanto, un solapamiento de turnos de 15h a 17h en los días lunes a sábado.

2.2.3 Restricciones

La primera restricción del problema es que los cocineros no pueden trabajar más horas de las que tienen asignadas. La segunda restricción a tener en cuenta es que un cocinero no puede trabajar durante un turno de mañana y durante un turno de tarde del mismo día, puesto que dichos turnos se solapan. Una tercera restricción del problema, siendo esta una restricción blanda, es que los cocineros deberían librar dos días seguidos. De no ser el caso, la satisfacción de dicho empleado se divide a la mitad.

2.3 Subproblema: Camareros

2.3.1 Personal

El bar tiene a cinco camareros en plantilla, con disponibilidades variadas, con tres de ellos trabajando 40 horas y dos de ellos 20 horas, con disponibilidades y preferencias descritas en la tabla 2.

Cocinero	**Jornada**	**Disponibilidad**	**Preferencias**
Ca1	40h	Lunes a viernes por la mañana	Ninguna
Ca2	40h	Siempre	Mañanas
Ca3	40h	Solo turnos de 8h.	Tardes
Ca4	20h	Viernes tarde a domingo	Turnos de 4 horas
Ca5	20h	Viernes tarde a domingo	Ninguna

Tabla 2: Camareros en plantilla

2.3.2 Turnos

Existen un total de veintidós turnos individuales para camareros, divididos en trece turnos de ocho horas, siete de ellos de 9h a 17h (lunes a domingo) y seis de ellos de 17h a 01h (lunes a sábado), y además nueve turnos de cuatro horas, cinco de ellos de 12h a 16h (miércoles a domingo) y cuatro de ellos de 19h a 23h (miércoles a sábado). Los turnos de cuatro horas son turnos de refuerzo, que se solapan cada uno con un turno de ocho horas.

2.3.3 Restricciones

En esta instancia, además de las restricciones de que los camareros no puedan trabajar durante más horas de las asignadas, y de que su satisfacción se reduce a

la mitad si no pueden librar durante dos días seguidos, se presentan las siguientes restricciones: un camarero no puede trabajar en un turno de refuerzo que se solape con un turno de ocho horas, y no puede ser asignado a trabajar tampoco en un turno para el que no esté disponible.

3 Optimización con Opt4J

3.1 Herramienta: Opt4J

Opt4J [1] es una herramienta para computación basada en mecanismos evolutivos, de código abierto y basada en Java, que contiene algoritmos de optimización multi-objetivo para algoritmos evolutivos, evolución diferencial, optimización de enjambre de partículas y enfriamiento simulado.

La herramienta requiere que el problema se modele en un archivo *Creator*, que define el genotipo del problema, un archivo *Decoder*, que pasa del genotipo al fenotipo, y un archivo *Evaluator*, que evalúa el fenotipo. Se añade a esto, además, un archivo con los datos del problema.

3.2 Representación de los datos

Para la correcta representación del problema se han creado las siguientes estructuras:

3.2.1 Cocina

- Constantes *numTurnosCocina* y *numCocineros*, representando respectivamente el número de turnos de cocina y el número de cocineros.
- Array *disponibilidadCocineros*, de dimensiones *numCocineros* x *numTurnosCocina*, que representa mediante valores numéricos la disponibilidad de los cocineros y su preferencia, siendo 1 disponible y 2 un turno por el que el cocinero tiene preferencia.
- Array *horasCocineros*, de dimensión 1 x *numCocineros*, indicando las horas que tiene disponibles cada cocinero.
- Array *duracionTurnosCocina*, de dimensión 1 x *numTurnosCocina*, indicando la duración de cada turno.

3.2.2 Camareros

- Constantes *numTurnos* y *numCamareros*, representando respectivamente el número de turnos de camareros y el número de camareros.

[1] https://sdarg.github.io/opt4j/

- Array *disponibilidadCamareros*, de dimensiones *numCamareros* x *numTurnos*, representando mediante valores numéricos la disponibilidad y preferencia de los cocineros. Un valor positivo indica disponibilidad, 0 indica que el camarero no está disponible, y valores mayores a 1 indican preferencia.
- Array *horasCamareros*, de dimensión 1 x *numCamareros*, indicando las horas que tiene disponibles cada camarero.
- Array *duracionTurnos*, de dimensión 1 x *numTurnos*, indicando la duración de cada turno.

3.3 Creator

El archivo Creator sigue una estructura similar en ambas instancias: debido a nuestro objetivo, que es la optimización de una instancia ya existente, se crea un genotipo de tipo *IntegerGenotype* y tamaño número de turnos, con rango de valores de 1 al número de empleados de la instancia. Posteriormente, este se inicializa a los valores de un genotipo ya factible, distinto para Cocina y Camareros y previamente calculado, para que este sea el origen de la población de soluciones.

3.4 Decoder

El archivo *Decoder* también se comporta de la misma manera en ambas instancias: pasa de un genotipo a un fenotipo en forma de lista de tipo *ArrayList*, conteniendo cada uno de los elementos del genotipo en el orden en que aparecían en este.

3.5 Evaluator

3.5.1 Cocina

Para el cálculo de la *fitness* del individuo, primeramente se comprueba mediante un bucle anidado si los cocineros trabajan un número de horas adecuado. De no ser el caso, el individuo se evalúa al valor mínimo entero que Java permite. Seguidamente se comprueba en un segundo bucle tanto que el cocinero no realice un turno de mañana y su correspondiente turno de tarde, como que esté disponible en el horario adecuado. De nuevo, de no cumplirse estas restricciones, se devuelve el valor mínimo posible. Finalmente se comprueba si el cocinero libra dos días seguidos y se organiza esta información como un valor booleano en un array, calculando posteriormente el valor de satisfacción obtenido. De no librar dos días, la satisfacción obtenida por dicho cocinero se reduce a la mitad.

3.5.2 Camareros

En la instancia de camareros, la *fitness* del individuo se calcula de forma bastante similar: se calcula primero si se trabaja el número de horas adecuado,

devolviendo un valor mínimo de no ser el caso, y posteriormente se comprueba que no se trabajen turnos que se solapan, siendo esta una comprobación más compleja que para cocina debido al modo en que se organizan los turnos.
Posteriormente se calcula la disponibilidad de los camareros, se comprueba si estos libran dos días seguidos y, tras esto, se utiliza esta información para calcular la satisfacción y, por tanto, la fitness del individuo.

3.6 Resultados

3.6.1 Genotipo inicial cocina

El problema parte de la configuración inicial expuesta en la tabla 3, en que todos los cocineros libran dos días seguidos pero generalmente no trabajan sus turnos preferidos, obteniendo una satisfacción de **15**. Se resuelve, además, mediante la herramienta de algoritmos genéticos de Opt4J y los parámetros por defecto.

3.6.2 Mejor resultados cocina

Tras un total de 1000 generaciones, así como cinco experimentos, la mejor satisfacción para la instancia de cocina se ha obtenido con el individuo descrito en la tabla 4, que obtiene un valor de **23**, reorganizando los turnos para cumplir mucho mejor con las preferencias de los cocineros

3.6.3 Genotipo inicial camareros

El problema parte de la configuración inicial expuesta en la tabla 5, en que todos los cocineros libran dos días seguidos pero generalmente no trabajan sus turnos preferidos, obteniendo una satisfacción de **30**. Se resuelve, de nuevo, mediante la herramienta de algoritmos genéticos de Opt4J y los parámetros por defecto.

3.6.4 Mejor resultados camareros

Tras un total de 1000 generaciones, así como cinco experimentos, la mejor satisfacción para la instancia de cocina se ha obtenido con el individuo descrito en la tabla 6, que obtiene un valor de **33**, mejorando levemente la satisfacción obtenida a base de reorganizar los turnos del Martes y del Sábado.

Turno	Empleado	Preferencia
LM	1	-
LT	2	-
MM	1	-
MT	2	-
MM	1	-
MT	2	-
JM	3	Sí
JT	2	-
VM	3	Sí
VT	2	-
SM	1	Sí
ST	3	Sí
DM	1	Sí

Tabla 3: Genotipo Inicial Cocina

Turno	Empleado	Preferencia
LM	2	Sí
LT	1	Sí
MM	2	Sí
MT	1	Sí
MM	2	Sí
MT	1	Sí
JM	3	Sí
JT	2	Sí
VM	3	Sí
VT	2	Sí
SM	3	Sí
ST	1	-
DM	1	Sí

Tabla 4: Mejor Individuo Cocina

Turno	Empleado	Preferencia
LM	1	-
LT	3	Sí
MM	1	-
MT	3	Sí
MM	1	-
MMr	2	-
MT	3	Sí
MTr	2	-
JM	1	-
JMr	2	-
JT	3	Sí
JTr	2	-
VM	1	-
VMr	2	-
VT	3	Sí
VTr	4	Sí
SM	4	-
SMr	5	-
ST	5	-
STr	4	Sí
DM	5	-
DMr	4	Sí

Tabla 5: Genotipo Inicial Camareros

Turno	Empleado	Preferencia
LM	1	-
LT	3	Sí
MM	2	Sí
MT	3	Sí
MM	1	-
MMr	2	-
MT	3	Sí
MTr	2	-
JM	1	-
JMr	2	-
JT	3	Sí
JTr	2	-
VM	1	-
VMr	2	-
VT	3	Sí
VTr	4	Sí
SM	2	Sí
SMr	4	Sí
ST	5	-
STr	4	Sí
DM	5	-
DMr	4	Sí

Tabla 6: Mejor Individuo Camareros

4 Problema de scheduling

Para la parte de scheduling tomamos el problema a resolver, organizar los turnos de los trabajadores en un bar, pero lo enfocamos de una manera distinta. En vez, optimizar la satisfacción de los trabajadores dadas sus preferencias el objetivo es desde cero crear horario para todos los trabajadores. De forma que dada la disponibilidad de cada trabajador, su horas de contrato laboral y las necesidades del bar se cree un horario para sus trabajadores.

La tarea como se ha comentado previamente se puede interpretar como una tarea de scheduling con 2 subproblemas, los cocineros y los camareros. Además, la cocina al tener menos trabajadores es una instancia menos compleja a resolver.

Cabe resaltar, que para esta resolución se presupone que hace falta más de un trabajador en los horarios de refuerzo, haciendo la instancia un poco más compleja a la resolución anterior. A su vez se ha simplificado una parte del problema tomando en cuenta la disponibilidad de un empleado como el día en general, no las mañanas y las tardes por separado.

4.1 Métricas

Para la resolución de este problema, solamente nos centraremos en su factibilidad e intentar reducir el tiempo de ejecución del problema. Por ello, al evaluar la modelización del problema, nos fijaremos en estos dos aspectos intentando que sea lo más generalizable posible.

4.2 Datos del problema

4.2.1 Personal

El bar tiene en su plantilla tres cocineros distintos los cuales están disponibles todos los días. Sin tener en cuenta sus preferencias, ya que esta resolución no las toma en cuenta, su información se encuentra en la Tabla 1.

Además como se ha comentado antes también cuenta en su plantilla con 5 camareros, los cuáles tienen distinta disponibilidad y jornadas. Dado que la disponibilidad se presenta como días enteros difiere un poco de la Tabla 2, pero la información de estos trabajadores es visible en la siguiente tabla.

Cocinero	**Jornada**	**Disponibilidad**
Ca1	40h	Lunes a viernes
Ca2	40h	Siempre
Ca3	40h	Siempre
Ca4	20h	Viernes domingo
Ca5	20h	Viernes domingo

Tabla 7: Camareros en plantilla

4.2.2 Turnos

Los turnos son generalizables para los dos tipos de trabajadores. Estos se dividen en:

- Día de descanso.
- Mañana completa.
- Media mañana.
- Tarde completa.
- Media tarde.
- Media mañana y media tarde.

4.2.3 Restricciones

De nuevo las restricciones son las mismas para ambos tipos de trabajadores. Las diferentes restricciones son:

- Sólo se puede trabajar cuando se esta disponible.
- No se puede trabajar más de 5 días a la semana o debe haber dos días de descanso.
- El número de horas trabajadas debe ser igual a las de su contrato.
- Los trabajadores de un turno deben ser iguales o superiores a la necesidad del mismo.

5 Scheduling con MiniZinc

5.1 Herramienta: MiniZinc

Minizinc [2] es un lenguaje de modelado para expresar información, conocimiento o sistemas de problemas de inteligencia artificial. Dado un conjunto de reglas definidas, estas se usan para la interpretación de los componentes del sistema.

5.2 Representación de los datos

Para la representación de este problemas utilizamos diversas variables constantes y variables de decisión a resolver por el resolutor.

Hay constantes que por el nombre son auto-explicativas como *days*, *work_days*, *num_chefs* o *num_chefs*. Pero a continuación se explicaran el resto. La variable *turns* hace referencia al número de turnos distintos de los trabajadores. En este problema tenemos 4 distintos, la mañana completa, media mañana, la tarde

[2] https://www.minizinc.org/

completa y media tarde. El array *working_hours* hace referencia al número de horas de los turnos de trabajo. Estos son trabajar media o una mañana completa, media o una tarde completa o media mañana y media tarde. Siendo 8, 4, 8, 4 y 8 horas respectivamente.

Otras variables constantes son la necesidad de trabajadores, representada en un array de los distintos turnos para una semana completa. También es necesario un array bidimensional de la disponibilidad diaria de cada trabajador y por último un array de cuantas horas laborales tiene cada trabajador. Estos tres elementos son necesarios para cada tipo de trabajador. Es puede hacer referencia a ellos como *workers_needed*, *workers_available* y *workers_contract*.

Finalmente, es necesaria la variable de decisión para generar el horario de la semana del bar. La variable *workers_timetable* especifica con 6 valores distintos a cada trabajador su turno de trabajo para cada día de la semana.

- 0: Descanso
- 1: Mañana completa
- 2: Media mañana
- 3: Tarde completa
- 4: Media tarde
- 5: Media mañana y media tarde

5.3 Representación de las restricciones

5.3.1 Sólo se puede trabajar cuando se esta disponible

Se comprueba que todos los trabajadores tienen se les ha asignado días en los que están disponibles comparando *workers_timetable* y *workers_availability*.

5.3.2 No se puede trabajar más de 5 días a la semana o debe haber dos días de descanso

Para todos los trabajadores se comprueba que la suma de sus turnos es menor o igual que *work_days*.

5.3.3 El número de horas trabajadas debe ser igual a las de su contrato

Para cada trabajador la suma de las referencias en *working_hours* en los turnos asignados en *workers_timetable* debe ser menor que su *workers_contract*.

5.3.4 Los trabajadores de un turno deben ser iguales o superiores a la necesidad del mismo

Se hace un conteo de los trabajadores para cada turno y se comprueba si esta suma es mayor o igual a *workers_needed*.

5.4 Resultados

Dada la instancia inicial se ha comprobado que no existe solución para las restricciones de los camareros. Debido a esto se ha introducido un camarero con total disponibilidad para simular que ha sido contratado.

Se han lanzado 10 veces cada experimento y se ha hecho una media del tiempo para conseguir resolver la instancia. La siguiente tabla muestra los resultados obtenidos.

Instancia	**Tiempo (s)**
Sólo cocineros	0.35
Sólo camareros	UNSATISFIABLE
Sólo camareros + 1	0.39
Todos los trabajadores + 1	135.47

Tabla 8: Camareros en plantilla

Se puede observar dados los resultados anteriores, es mucho más preferible resolver los subproblemas de los cocineros y camareros por separado ya que reduce la complejidad de la instancia de tal forma que el tiempo de computo en 100 veces menor. Además podemos comprobar como el problema se vuelve factible una vez se añade un camarero extra a la instancia. De esta forma se pueden resolver todas las necesidades de los turnos de la semana.

6 Conclusiones

Se ha procedido a detallar el problema, enfocado de dos formas distintas con dos herramientas distintas, y a exponer el diseño de la solución y resultados, obteniendo pues resultados factibles que nos permiten alcanzar el objetivo buscado en este trabajo: la optimización y organización de los turnos de un bar.

Se ha utilizado pues una primera herramienta (Opt4J) para buscar la optimalidad de los turnos según la métrica decidida, y seguidamente, para mayor completitud, se ha utilizado una segunda herramienta (MiniZinc) para buscar una mejor organización de los turnos partiendo desde cero, y obteniendo resultados factibles.

6.1 Dificultades encontradas

Durante el desarrollo del proyecto se han encontrado diversas dificultades, entre ellas la búsqueda y selección de herramientas adecuadas al problema y, quizás la más importante, el tiempo de cómputo necesario para obtener una solución factible desde cero.

6.2 Trabajos futuros

6.2.1 Optimización

Como posible trabajo futuro de optimización sería interesante considerar tanto poder partir de distintos genotipos factibles, creándolos en el archivo *Creator* con un mayor componente aleatorio, así como también sería de utilidad modificar el problema para añadir turnos de refuerzo con más empleados por turno.

6.2.2 Scheduling

Para la parte de scheduling sería interesante como trabajo futuro especificar la disponibilidad de los trabajadores como mañana y tarde en vez de por días completos y poder simular la contratación de nuevos empleados para así satisfacer instancias propuestas imposibles.