Asignación de viviendas en cooperativas: Programación por restricciones y Análisis de equidad

Informe de Proyecto de Grado presentado por

Marcos Fierro Ghittino

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisor

Héctor Cancela

Montevideo, 24 de junio de 2024

Agradecimientos

Quisiera agradecer especialmente a Héctor Cancela, por el gran soporte brindado durante la realización de este trabajo, por sus comentarios y sugerencias vinculadas a mejorar este informe, pero sobre todo por aquellas en el marco más amplio que es la profesión en sí. A todos los responsables de haber ideado y construido MTAV como aplicación y haberla difundido para lograr su uso real en la sociedad. Finalmente a mis familiares - Jessica, Mariella, Leo, Guille, Paola y Kate - así como a mis amigas y amigos por su apoyo constante y consideración.

Resumen

El presente trabajo de fin de grado profundiza en la optimización de una aplicación informática adaptada a cooperativas de vivienda. El objetivo principal es asignar viviendas de manera eficiente a los usuarios en función de sus preferencias individuales. Estas preferencias forman la base de un complejo proceso de emparejamiento, en el que la aplicación considera y equilibra meticulosamente los deseos de cada usuario con las opciones de vivienda disponibles.

Inicialmente, la tesis examina una implementación de programación matemática existente diseñada para este propósito. A través de una evaluación comparativa, su desempeño se evalúa frente a un nuevo enfoque de Constraint Programming propuesto como parte de la tesis y desarrollado haciendo uso del lenguaje MiniZinc (open-source). Este análisis comparativo arroja luz sobre las fortalezas y debilidades de cada método y, en última instancia, orienta la selección de la estrategia o herramienta más eficaz para la asignación de viviendas dentro de las cooperativas de vivienda.

Además, se introduce una dimensión extra en el proceso de asignación al incorporar un concepto de equidad junto con las métricas de satisfacción tradicionales. En este contexto, la solución no sólo busca maximizar la satisfacción general de los usuarios, sino que también explora la equidad en la asignación de viviendas. La métrica de equidad adoptada para la evaluación es la desviación estándar, que ofrece una medida cuantitativa de cuán equitativamente se distribuyen las viviendas entre los usuarios en términos de satisfacciones individuales. En un capítulo subsiguiente, el trabajo expone una prueba de concepto relativa a la integración de un sistema de preferencias paralelo vinculado a los deseos de los usuarios por ser vecinos de otros usuarios en específico, debido por ejemplo a su vínculo familiar o de amistad. Al contemplar esta entrada de datos adicional, el nuevo modelo investiga cómo el cumplimiento de esta condición puede mejorar los niveles de satisfacción individual y así una distribución de viviendas que quizás antes no resultaba relevante para el sistema, puede volverse óptima tras ser valorada de forma más precisa. Este enfoque responde a una sugerencia previamente obtenida por parte de los usuarios en instancias de uso reales y pretende brindar mayor importancia a las conexiones sociales reconociendo la dinámica comunitaria en las cooperativas de vivienda.

En resumen, la tesis da un primer paso sobre nuevos aspectos no contemplados hasta el momento en la problemática de las asignaciones en cooperativas obteniendo conclusiones preliminares sobre la viabilidad y utilidad de cada aspecto. A raíz del avance logrado y de MTAV como herramienta, se visualizan nuevas oportunidades de trabajo futuro tanto en la "usabilidad"de la aplicación como en el refinamiento y posterior validación junto a usuarios reales de las nuevas características analizadas en la solución.

Palabras clave: Asignación de Viviendas, Cooperativas, House Allocation, Housing Allocation, Constraint Programming

Índice general

Capítulo 1

Introducción

1.1. Motivación

Las cooperativas de vivienda son organizaciones formadas por un grupo de personas con el objetivo común de obtener cada una su vivienda asequible y satisfacer sus necesidades habitacionales. Alternativamente a comprar una vivienda de manera individual, los miembros de una cooperativa de vivienda se unen para adquirir y administrar una propiedad colectivamente. Generalmente la construcción del complejo entero es realizada - bajo guía y supervisión profesional - por los propios miembros de la cooperativa. Realizando tal esfuerzo, se logran reducir en gran manera los costos generales de la adquisición de una solución habitacional para las personas.

En una cooperativa, son dichos miembros los usuarios de las viviendas. Cada socio tiene derecho a utilizar una unidad residencial en la propiedad de la cooperativa y participa activamente en la toma de decisiones relacionadas con la administración y el mantenimiento de la cooperativa. Las cooperativas de vivienda funcionan bajo los principios de ayuda mutua, participación democrática, responsabilidad individual y gestión colectiva. Este modelo ha sido utilizado en países como Canadá, Suecia, Polonia, Noruega y España (Housing Europe, 2012) además de también en la mayoría de países de América Latina (Sánchez-Laulhé Sánchez de Cos, Rabasco Pouzelo, y Solanas Domínguez, 2013). En Uruguay es quizás donde puede considerarse que este modelo ha tenido mayor desarrollo, tanto que desde 1970 (FUCVAM, s.f.) hasta la fecha de hoy, las cooperativas son y han sido una solución frecuente dentro de la clase media donde el acceso a la vivienda muchas veces no es algo particularmente sencillo de lograr.

Culminado el gran trabajo de construir desde los cimientos un complejo de viviendas, uno de los momentos finales más ansiado es aquel en el que cada núcleo familiar conocerá la unidad en la que vivirá. Esta asignación no es en absoluto menor, ya que determina la unidad que cada familia tendrá por hogar, a la vez que también es una instancia única, significando que las mencionadas asignaciones no vuelven a tener lugar posteriormente (o al menos no es nada común que se repitan).

El procedimiento más habitual para asignar las viviendas es el sorteo aleatorio, que no permite tener en cuenta ningún tipo de preferencias por parte de los cooperativistas ni tampoco ninguna distinción entre los mismos. Otro mecanismo menos frecuentado es la selección por antigüedad, en donde, como intuye el nombre, los miembros tienen la posibilidad de seleccionar las viviendas según la antigüedad de su vínculo con la cooperativa. Estos casos son poco comunes aunque pueden tener lugar en cooperativas que por alguna razón su finalización se haya visto impedida y por lo tanto, cuando dicha situación se prolonga, algunos socios pueden optar por buscar una solución alternativa de vivienda y desvincularse de la asociación. En el año 2016 se desarrolló MTAV (Mejor Tecnología de Asignación de Viviendas), una herramienta para realizar dichas asignaciones a partir de preferencias indicadas por los cooperativistas. Esta herramienta, que ha sido desarrollada por un equipo de docentes, estudiantes y egresados de la Facultad de Ingeniería de la Universidad de la República, con participación también de cooperativistas, consiste en la aplicación de modelos de programación lineal entera para optimizar los resultados finales. Su uso ha sido muy bien valorado

por numerosas cooperativas. La experiencia surgida de la aplicación de MTAV ha permitido detectar a lo largo de las instancias, distintos posibles cambios y mejoras al mismo tiempo que se han generado también algunos interrogantes al respecto. El objetivo de este proyecto de grado es continuar y complementar el desarrollo de MTAV, partiendo de la experiencia ya existente.

Actualmente, de forma sencilla aunque igual eficaz, MTAV se basa únicamente en las prioridades de cada cooperativista sobre las viviendas disponibles en su complejo. Estas prioridades o preferencias que cada usuario ingresa son el dato fundamental para el resultado de esta aplicación. En ediciones anteriores de trabajos hechos por parte de estudiantes y docentes de la UDELAR (Prino, Sánchez, y Cancela, 2016) (Fagián, Prino, y Sánchez, 2017) (Fierro, 2020) se realizaron múltiples actualizaciones a la aplicación con el fin de facilitar su uso a los cooperativistas además de mejorar su experiencia final como usuarios. Entre los principales focos de mejora, dos de los más trabajados fueron el modo en que los usuarios ingresan sus preferencias, y la respuesta de la aplicación durante el sorteo (interacciones y presentación de los resultados).

En el contexto presentado, surge la oportunidad de una vez más analizar la solución que está implementada actualmente en MTAV y evaluar las características de las asignaciones retornadas, para determinar si hay lugar a siguientes mejoras en las mismas.
Esta tesis aborda el punto anterior con el objetivo de generar un aporte a la línea de trabajo de optimizar la calidad de las soluciones retornadas así como a la posibilidad de ampliar las capacidades del sistema en cuanto a la incorporación de distintos criterios de prioridad o ponderaciones que los usuarios puedan encontrar de utilidad a la hora de representar con mayor precisión sus necesidades.

Este documento expone las conclusiones obtenidos tras la investigación y posterior implementación de propuestas asociadas al concepto de equidad de una solución final y a la posibilidad de expandir los criterios de preferencias de los usuarios.

1.2. Objetivos Propuestos

Particularmente, el presente trabajo propone como objetivo central, investigar la calidad de las asignaciones encontradas desde un nuevo punto de vista: la equidad que éstas presentan para los distintos cooperativistas participantes. En este sentido, se buscará plantear una definición de equidad sobre las prioridades asignadas de tal manera que dicha definición sea cuantificable y por tanto sea posible llevar a cabo comparaciones de esa medida de equidad entre las diferentes soluciones factibles. Esto puede implicar nuevas etapas e iteraciones en el proceso de cálculo de la asignación que pueden no ser en un principio, sencillas de implementar utilizando únicamente modelos de programación matemática. La equidad en los resultados ha sido uno de los mayores focos de atención por parte de los usuarios cuando la asignación óptima retornada implica que uno o unos pocos cooperativistas estén notoriamente distanciados del resto en cuanto a sus preferencias, tanto sea el caso de beneficio propio como lo opuesto. Es fácil identificar que si una vivienda resulta favorita o la menos deseada para todos o la mayoría, en cualquier caso alguna familia vivirá en ella y entonces en los resultados alguien será el beneficiado o perjudicado. La oportunidad de investigación se encuentra en los casos menos intuitivos, aquellos que surgen de la propia naturaleza combinatoria del problema y que quizás, la solución con la que se cuenta no explora exhaustivamente para garantizar una equidad óptima a los socios de las cooperativas.

Por otra parte y teniendo en cuenta el objetivo anterior, se plantea también la búsqueda de un enfoque alternativo para abordar el problema de las asignaciones con el propósito de permitir la posibilidad de expandir las opciones que los usuarios tienen para definir sus criterios y/o preferencias sobre los posibles resultados de asignación de vivienda para sí mismos. Un ejemplo de ello sería el hecho de que los usuarios en ciertos casos podrían encontrarse beneficiados no solo por la vivienda asignada sino también por otro aspecto como lo son los vecinos directos (adyacentes o contiguos) que tendrán. Este caso puede ser una gran diferencia para muchos cooperativistas si estos presentan vínculos familiares o de amistad con algunos de los

miembros también participantes. El concepto de vecindad ha sido ocasionalmente consultado en casos reales de cooperativas que hicieron uso de MTAV y plantearon la opción. Similar a lo anterior, cualquier factor indefinido que pueda alterar dinámicamente las preferencias de los cooperativistas dada una misma vivienda es una entrada de datos ponderada que puede impulsar o desfavorecer ciertas combinaciones de asignaciones. Por ejemplo, en el hipotético caso en el que una cooperativa en un futuro fuese a incluir una estación de carga para vehículos eléctricos pero que por algún motivo, aún se desconoce si se instalará al frente o al fondo del complejo, en un caso como tal, el sistema podría facilitar tal incertidumbre y contemplar ambos casos de ubicaciones de la estación, ofreciendo una sugerencia de solución tras analizar si la satisfacción global sería mejor en un caso u otro.

Teniendo presentes los dos objetivos centrales mencionados, se considera de interés explorar un enfoque de resolución que sea versátil frente a cambios en el modelado y también en las búsquedas a realizar. La opción seleccionada para lo anterior es entonces Constraint Programming pues presenta una mayor flexibilidad de modelado y, en cuanto a la configuración de las estrategias de búsqueda, su parametrización es muy sencilla. Cabe mencionar que dentro de la metodología de MIO (Mixed Integer Optimization), la resolución de problemas cuadráticos es posible también, habiendo solvers específicos para esto con grandes capacidades pero incluso en casos sencillos donde los términos cuadráticos se encuentran solamente en la función objetivo y no en las restricciones, entonces solvers de carácter más genérico como GLPK, también suelen ser capaces de resolverlos.

1.3. Organización del Informe

El informe presentado tiene la siguiente estructura:
En el Capítulo 2 se detalla el contexto de la problemática asociada a este trabajo, la de las cooperativas de viviendas, y puntualmente vinculado a las mismas, la asignación de viviendas a los socios de la cooperativa. El capítulo comienza explorando la solución actual desarrollada por la Facultad de Ingeniería de la UDELAR continuando con un análisis sobre la misma para luego finalizar presentando el enfoque alternativo de trabajo escogido - Constraint Programming (CP) - para extender la mencionada solución. Este enfoque permite algunos niveles de configuración con relación a la búsqueda de soluciones y por tanto se discuten los principales parámetros y elementos de configuración que para este problema ofrecen una búsqueda con mayor eficacia y eficiencia.
En el capítulo 3 se presentan distintos modelos desarrollados con CP para el mismo problema que se ha resuelto por Mixed-Integer Programming (MIP). También se listan los casos de prueba seleccionados para utilizar en la ejecución de los modelos y se incluyen los resultados asociados a los mismos, comparándolos contra aquellos de la solución actual. Posteriormente se continúa con la extensión de la solución incorporando una propuesta de medida de equidad, la cual luego es también testeada con los mismos casos de prueba anteriores obteniendo nueva información a partir de los resultados experimentales.
Por último en el capítulo 4 se concluye con los comentarios finales, las conclusiones obtenidas a raíz del trabajo realizado y las propuestas de trabajo futuro asociadas a la idea de ofrecer la mejor experiencia y solución para las cooperativas a la hora de definir sus asignaciones.

Capítulo 2

Antecedentes y Metodologías de interés

En el presente capítulo se exponen los problemas de asignación en general como principal área de estudio de este trabajo. Dentro de los problemas de asignación se encuentran los vinculados a contextos de viviendas. Se presenta MTAV, la herramienta que ha cambiado la forma en que las viviendas son asignadas dentro de las cooperativas y se discute el uso de Constraint Programming como paradigma alternativo que posteriormente se utilizará para extender tanto el análisis de los resultados de MTAV como las capacidades de dicha aplicación.

2.1. Problemas de asignación

Los problemas de asignación (también referenciados como *minimum weight perfect matching problems in bipartite graphs*) (Korte, Vygen, Korte, y Vygen, 2011) son una clase de problemas de programación lineal bien conocidos del campo de la optimización combinatoria u optimización discreta (búsqueda de subconjuntos que sean las mejores soluciones dentro de un conjunto finito dado). Estos problemas, los cuales en algunos casos pueden presentarse también como *network flow - max flow problems* (Wheeler, Wei, Bahadir, Shui, y Zhang, 2020), se encuentran entre los más estudiados y aplicados debido a su relevancia en una amplia gama de aplicaciones prácticas como la organización de eventos en salas, tareas entre personas, personas a viviendas, etc.

En su forma más básica, el problema consiste en asignar un número de elementos a un número de destinos de modo de optimizar un beneficio objetivo, ya sea minimizar tiempos, maximizar ganancias, etc. dependiendo de qué representen dichos elementos y destinos de asignación. Una instancia de enunciado muy común de este problema es la siguiente:

"Dado un número de agentes y un numero de tareas donde cualquier agente puede realizar cualquiera de las tareas bajo un costo fijo el cual difiere dependiendo tanto del agente como de la tarea, se busca minimizar dichos costos asociados de ejecución asignando siempre un solo agente a una tarea y a lo sumo una tarea para cada agente."

Dentro de las muchas variantes de este tipo de problemas, se encuentran distintas dificultades y costos computacionales al momento de resolverlos. Aunque las instancias más simplificadas de problemas de asignación se categorizan como problemas P (problemas para los cuales se conoce un algoritmo capaz de resolverlos en tiempos polinomiales), una vez se expande la realidad presentada y se incorporan más aspectos de la misma, el problema se vuelve notoriamente más complejo. Existe una formulación genérica de este tipo de problemas (GAP - Generalized Assignment Problem) y es clasificada como NP-hard (Michael R. Garey,

1979) (Özbakir, Baykasoğlu, y Tapkan, 2010) pues añade conceptos y restricciones adicionales. A modo de ejemplo se puede considerar la siguiente instancia del GAP:

"Tenemos N objetos y M transportes. Los objetos tienen un beneficio al ser trasladados en los distintos transportes y tienen un volumen de ocupación asociado. Los transportes tienen asociado también un espacio máximo de capacidad y un costo por llevar cada una de los distintos objetos. Luego, el problema que se presenta es el de maximizar las ganancias obtenidas por traslados de objetos utilizando los distintos transportes sin poder sobrecargar sus capacidades dadas."

Notar que la diferencia en este caso es la necesidad de considerar, aparte de costos, los nuevos parámetros de espacio tanto los requeridos como los máximos disponibles.

Si bien existen varios algoritmos para resolver problemas lineales en general, aplicar algoritmos genéricos que no toman ventaja de las características específicas de un problema de asignación resulta, en la mayoría de los casos, menos eficiente que algoritmos especiales que se beneficien de la estructura del problema.(Korte y cols., 2011) Tal como se mencionó al comienzo, es posible visualizar o plantear el problema como un problema de flujo en grafos. En una instancia simple, puede ser un grafo bipartito ponderado en donde el objetivo es encontrar una bisección cuya suma de valores asociados a las aristas sea óptima. Por lo tanto, los algoritmos con mayor eficacia para la resolución de estas realidades son aquellos provenientes de la rama de investigación conocida como teoría de grafos. Ejemplos de algoritmos conocidos son el *Método Húngaro* (Kuhn, 1955) o el *Successive Shortest Path* (Engquist, 1980).

La formulación para una versión polinomial y para el GAP se detallan a continuación:

Parámetros:

n : Número de tareas.
m : Número de agentes.
c_{ij} : Costo de asignar el agente i a la tarea j.

Variables de Decisión:

x_{ij} : Variable de decisión binaria que indica si el agente i es asignado a la tarea j.

Problema de asignación Polinomial

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} \cdot x_{ij} \tag{2.1}$$

$$\text{Subject to} \quad x_{ij} \in \{0,1\}, \quad \forall i \in \{1,2,...,n\}, \forall j \in \{1,2,...,m\} \quad \textit{Binary Constraint} \tag{2.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \forall j \in \{1,2,...,m\} \quad \textit{One task per agent} \tag{2.3}$$

$$\sum_{j=1}^{m} x_{ij} = 1, \quad \forall i \in \{1,2,...,n\} \quad \textit{One agent per task} \tag{2.4}$$

Pasando (2.4) a la versión generalizada del problema de asignaciones, se mantienen las variables binarias de decisión y se añaden dos nuevos parámetros vinculados a capacidades:

Parámetros:

n : Número de objetos.
m : Número de transportes.
c_{ij} : Costo de trasladar el objeto i en el transporte j.
e_{ij} : Espacio ocupado por el objeto $i$al trasladarlo en el transporte j.
d_j : Disponibilidad de espacio del transporte j.

Generalized Assigned Problem - GAP

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij} \cdot x_{ij} \tag{2.5}$$

$$\text{Subject to} \quad \sum_{i=1}^{n} e_{ij} \cdot x_{ij} \leq d_j, \quad \forall j \in \{1,2,...,m\} \quad \textit{Space Capacity Constraint} \tag{2.6}$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in \{1,2,...,n\}, \forall j \in \{1,2,...,m\} \quad \textit{Binary Constraint} \tag{2.7}$$

$$\sum_{j=1}^{m} x_{ij} \;=\; 1, \quad \forall i \in \{1,2,...,n\} \quad \textit{One transport per object} \tag{2.8}$$

Housing Allocation

De las múltiples variaciones del problema de asignación, el problema con el nombre de "Housing Allocation" es quizás uno de los más similares al abordado en este trabajo. Según (Long, 2016) "Desde el trabajo pionero de Shapley y Scarf (1974), una gran cantidad de literatura ha estudiado problemas de asignación de recursos discretos. Un problema de este tipo está formado por un conjunto de agentes y un conjunto de objetos indivisibles. Cada agente consume un objeto pero tiene una relación de preferencias sobre el conjunto de objetos conocida únicamente por él mismo. Cada objeto sólo se puede asignar a un agente, y a los objetos no les importa a quién están asignados. Estos problemas son conocidos como problemas de emparejamiento unilateral (one-sided matching) en contraposición a problemas de emparejamiento bilaterales (two-sided matching)... Cuando Shapley and Scarf (1974) formalizaron el problema, utilizaron un ejemplo de hogares y comerciantes. Desde entonces, estos problemas se conocen ahora como ***house allocation*** problems."

Existe una variante ligeramente diferente muy conocida en la literatura también llamada ***Housing Market***, donde los agentes inicialmente poseen un hogar propio y pueden opcionalmente intercambiarlo con otro dentro del mercado. [1]

Como se mencionó anteriormente, la interpretación simplificada (preferencias one-sided de usuarios sobre unidades) de la realidad abordada en las cooperativas de viviendas y también en este informe coincide en

[1] Housing Market problem - https://en.wikipedia.org/wiki/House_allocation_problem

sus características con la descripción del problema "Housing Allocation" ya que no se cuenta con posesiones iniciales de hogares y tampoco hay intercambios voluntarios o consensuados de hogares entre miembros durante la asignación. Luego al estudiar la expansión de la solución aplicada al problema, una vez se incluyen otros parámetros de entrada diferentes a las preferencias de vivienda, el problema se diferencia del caso Hosuing Allocation base y se torna una versión más compleja.

2.2. MTAV

Solución conceptual actual de MTAV

Actualmente, tal como se expone en el trabajo inicial de MTAV, en su informe final (Prino y cols., 2016), la herramienta emplea una búsqueda de asignaciones que consta de dos etapas, cada una con un objetivo diferente, y que en conjunto permiten hallar asignaciones de viviendas mucho más satisfactorias que aquellas pautadas siguiendo métodos convencionalmente utilizados tales como el sorteo aleatorio o variantes del mecanismo Serial Dictator [2].
La primera de las etapas mencionadas tiene como función objetivo maximizar la satisfacción individual de aquel cooperativista que resulte con el peor valor de prioridad asignado en relación al resto. Dicho de otra manera más intuitiva, busca aquellas asignaciones en las cuales la prioridad más alta utilizada (por tanto la peor prioridad escogida para alguien) sea lo menor posible.

Parámetros:

N : Número de cooperativistas.
H : Número de viviendas.
Asumimos $(N = H)$
p_{ij} : Preferencia del cooperativista i sobre la vivienda j.

Variables de Decisión:

x_{ij} : El cooperativista i es asignado a la vivienda j.

$$\begin{aligned}
&\text{Minimize} && Z \\
&\text{Subject to} && \sum_{j \in H} p_{k,j} x_{k,j} \ \leq \ Z, \ \forall k \in N && (2.9) \\
& && \sum_{j \in H} x_{k,j} = 1, \ \forall k \in N && (2.10) \\
& && \sum_{k \in N} x_{k,j} = 1, \ \forall j \in H && (2.11) \\
& && x_{k,j} \in \{0,1\}, \ z \in \mathbb{N}
\end{aligned}$$

Figura 2.1: Modelo de la primera etapa

[2]Serial Dictator - https://en.wikipedia.org/wiki/Random_priority_item_allocation/

Utilizando la información obtenida de la etapa 2.1 - el valor de cota superior de prioridades individuales hallado (llámese Z) - se procede con la ejecución de la etapa final cuyo objetivo es buscar las asignaciones que cumplan con Z entre sus prioridades y, de entre las mismas, buscar aquella con el menor valor agregado de prioridades asignadas que significa la mejor satisfacción grupal posible.

$$\text{Minimize} \quad \sum_{k \in N,\ j \in H} p_{k,j} x_{k,j} \tag{2.12}$$

$$\text{Subject to} \quad \sum_{j \in H} x_{k,j} = 1,\ \forall k \in N \tag{2.13}$$

$$\sum_{k \in N} x_{k,j} = 1,\ \forall j \in H \tag{2.14}$$

$$\sum_{j \in H} p_{k,j} x_{k,j} \ \leq \ Z,\ \forall k \in N \tag{2.15}$$

$$x_{k,j} \in 0, 1$$

Figura 2.2: Modelo de la etapa final

De esta manera describiendo MTAV como un todo, el mismo es entonces un mecanismo que maximiza lexicográficamente dos criterios bien diferentes. Ambos criterios de indiscutible utilidad para lograr una asignación eficiente de viviendas y siendo estos dos definidos respectivamente según su orden como la peor satisfacción individual permitida y la satisfacción acumulada o grupal.

2.3. MTAV - Implementación

Dado que MTAV incluye dos etapas de búsquedas distintas, así mismo se cuenta con dos modelos de programación lineal entera que implementan ambas etapas descriptas en la sección anterior. El resultado de la primer etapa es tomado como entrada en la ejecución posterior.

Una aclaración sobre los modelos implementados es que se asume que las cantidades de núcleos familiares y viviendas dentro de las cooperativas coinciden. Puede haber casos donde hayan viviendas que fuesen a quedar temporalmente vacías pues no hay quien las ocupe. En tales casos (los ha habido en algunas ocasiones reales de uso de MTAV) una rápida y sencilla solución es añadir un socio ficticio cuyas preferencias son todas iguales, concluyendo en que su asignación es indiferente y el algoritmo entonces no lo considerará clave o relevante en la toma de decisiones durante la búsqueda de soluciones. En el Listing 2.1 se muestra el modelo para la primer etapa y en el Listing 2.2 se presenta el modelo para el hallazgo de las asignaciones finales.

Modelo Intermedio - Búsqueda de la cota Z

```
1 /* #cooperativistas = #viviendas */
2 set N;
3
4 /* Prioridades de los cooperativistas */
```

```
param p{c in N, v in N};


/* Matriz de asignaciones */
var asig{n in N, v in N}, binary;

/* Variable auxiliar para representar la minima satisfaccion */
var z, integer;


minimize resultado: z;

s.t. z_menorIgual{n in N}: z >= sum{v in N} p[n,v] * asig[n,v];
/* z menor o igual que la satisfaccion de cada cooperativista n */

s.t. unicaAsignacionCoperativista_mayorIgual{n in N}:
    sum{v in N} asig[n,v] >= 1;
s.t. unicaAsignacionCoperativista_menorIgual{n in N}:
    sum{v in N} asig[n,v] <= 1;
/* el cooperativista n solo tiene una vivienda asignada */

s.t. unicaAsignacionCasa_mayorIgual{v in N}: sum{n in N} asig[n,v] >= 1;
s.t. unicaAsignacionCasa_menorIgual{v in N}: sum{n in N} asig[n,v] <= 1;
/* la vivienda v solo tiene un cooperativista asignado */

end;
```

Listing 2.1 Codigo Fuente - GNU MathProg - Cota Z

Modelo Final - Búsqueda de Satisfacción Global

```
/* #cooperativistas = #viviendas */
set N;

/* Prioridades de los cooperativistas */
param p{n in N, v in N};

/* Cota superior de preferencia para un socio */
param Z;

/* Matriz de asignaciones */
var asig{n in N, v in N}, binary;


/* Preferencias acumuladas */
minimize s: sum{n in N, v in N} p[n,v] * asig[n,v];

s.t. z_menorIgual{n in N}: sum{v in N} p[n,v] * asig[n,v] <= Z;
/* cota superior de preferencia para un socio */
```

```
19
20 s.t. unicaAsignacionCoperativista_mayorIgual{n in N}:
21     sum{v in N} asig[n,v] >= 1;
22 s.t. unicaAsignacionCoperativista_menorIgual{n in N}:
23     sum{v in N} asig[n,v] <= 1;
24 /* el socio n solo tiene una vivienda asignada */
25
26 s.t. unicaAsignacionCasa_mayorIgual{v in N}: sum{n in N} asig[n,v] >= 1;
27 s.t. unicaAsignacionCasa_menorIgual{v in N}: sum{n in N} asig[n,v] <= 1;
28 /* la vivienda v solo tiene un nucleo asignado */
29
30 end;
```

Listing 2.2 Codigo Fuente - GNU MathProg - Satisfaccion Global

La resolución de estos modelos no suele tardar más de unos pocos segundos en finalizar la ejecución de ambas etapas (lo cual es incluso muy rápido para la expectativa de los usuarios muchas veces). Sin embargo han surgido algunas ocasiones donde es necesario colocar un límite de tiempo a la primera etapa (el hallazgo de la cota Z) para evitar que el sistema se estanque sin avanzar en el proceso.

Este enfoque es el utilizado hasta la fecha en la búsqueda de asignaciones con la aplicación MTAV. La aplicación actual fué implementada como un programa de escritorio stand-alone ya que una vez instalado en el sistema el mismo incluye la herramienta GLPK (solver con el cual se ejecutan los modelos) y los demás componentes de software necesarios. [3]

2.4. MTAV - Análisis

Teoría económica - Análisis sobre MTAV y sus propiedades

Un trabajo muy interesante y directamente vinculado a MTAV es el realizado por Joaquín Paleo en su tesis (Paleo Arrarte, 2021) sobre las características que MTAV tiene como método que aborda el problema de housing allocation. En su investigación, propone analizar "...las propiedades que tiene este nuevo mecanismo desde la óptica de la teoría económica, las cuales generalmente se dividen en tres dimensiones: i) eficiencia, ii) compatibilidad de incentivos, y iii) equidad" (Paleo Arrarte, 2021).

A lo largo del informe, Paleo categoriza MTAV según distintas propiedades *ex-ante* y *ex-post* demostrando en cada caso las pruebas o ejemplos correspondientes para asegurar el cumplimiento o la falta de cada una de las mismas.

Como concluye en su trabajo, se puede corroborar que MTAV es **eficiente ex-post**, pues garantiza con total certeza que el resultado es Pareto-óptimo, indicando que no hay manera de encontrar una solución distinta en la que una familia obtenga una mejor asignación sin que otra resulte perjudicada. Continua el análisis desarrollando la proposición de **strategy-proof** la cual implica que para un método de resolución dado, no existe oportunidad alguna en la que, un individuo con parcial o total conocimiento de los datos entregados por los demás involucrados al problema, pueda tomar ventaja de esa situación mediante la manipulación de sus entradas alterándolas con respecto a las verdaderas (aquellas que hubiera definido si no tuviera la oportunidad mencionada). Haciendo uso de un claro y conciso ejemplo, demuestra que MTAV no siempre cumple con esta propiedad debido a que existen casos en los cuales un participante logra obtener mayor beneficio en el resultado si sus preferencias son modificadas correspondientemente para sacar ventaja frente a otros. Finaliza este estudio de propiedades verificando que MTAV sí satisface las condiciones de tener

[3] MTAV software - https://drive.google.com/file/d/13Hg2JpkD2OGQ7gt4NukXZzdFj5rUXZRS/view?usp=drive_link

equal treatment of equals, dicho de otra manera, presentar **equidad ex-ante** lo cual garantiza que si dos participantes de la asignación tienen exactamente las mismas preferencias sobre alguna vivienda, entonces también las probabilidades de adquirir esa vivienda son iguales para ambos dos, y debido a que MTAV aleatoriza la decisión frente a estos "empates.ºrdenando aleatoriamente en cada ejecución las filas de la matriz de preferencias, entonces la gestión de las prioridades de los cooperativistas es justamente equitativa ya que en cada iteración, los resultados de empate difieren.

Estas afirmaciones con sus respectivas demostraciones son clave para conocer MTAV como herramienta ya que permiten garantizar y destacar los puntos fuertes de la solución como también detectar de forma precisa los aspectos con posible vulnerabilidad. Esto último resultando esencial para planificar correctamente próximas mejoras y sacar el mayor valor de las mismas.

2.5. Constraint Programming

La programación basada en restricciones o Constraint Programming (CP) (de Harder, 2023) (IBM, 2022) es un paradigma que se caracteriza por describir problemas en términos de restricciones sobre las variables asociadas. En lugar de definir un algoritmo paso a paso rutinario (Najman, 2023) para la resolución de un problema, CP se centra en establecer relaciones lógicas entre variables y especificar las restricciones que deben cumplirse para que una solución sea calificada como válida.

Esta clase de programación se basa principalmente en fundamentos de ciencias de la computación como la programación lógica y la teoría de grafos, diferente a la programación matemática que se construye sobre álgebra numérica lineal. Las restricciones definen las condiciones y limitaciones entre los elementos del problema representados mediante variables, con lo cual una vez procesadas, las restricciones y sus implicancias definen nuevos subdominios para cada variable. El conjunto de potentes herramientas disponibles dentro de Constraint Programming es amplio; permite describir con gran detalle las particularidades de un problema dado. Entre muchas otras, algunas de estas opciones de modelado son *Option Types* (Mears y cols., 2014), *Interval Expressions*[4] y *Global Constraints*[5]. Como parte de este trabajo se analizarán también otras capacidades que los lenguajes de Constraint Programming ponen a disposición; capacidades de optar por distintos criterios a seguir durante la búsqueda en el espacio de soluciones. Estas opciones, dependiendo del problema a resolver, pueden generar significativas diferencias en la eficiencia de las ejecuciones y sus resultados (Çakır, Subulan, Yildiz, Hamzadayı, y Asılkefeli, 2022).

Considerando las posibilidades mencionadas de este enfoque, es claro que el mismo proporciona una forma flexible y declarativa de describir realidades, facilitando el modelado y la resolución de situaciones complejas (CP es una alternativa que refuerza sus fortalezas y ventajas cuanto más complejo es el problema).

Se aplica en una amplia variedad de grupos de problemas vinculados a la planificación logística, la asignación de recursos o la programación de horarios, entre otros. Un ejemplo de uso de este paradigma es el presentado en el artículo "A constraint programming approach for scheduling repetitive projects with atypical activities considering soft logic"(Zou y Zhang, 2020). En el mismo se desarrolla un modelo de CP que cubre la planificación de proyectos con actividades que constan de distintas unidades de trabajo, posiblemente ubicadas en distintas locaciones, y que pueden ser ejecutadas en paralelo o en reversa o ambas combinadas (a lo que llaman *soft* logic) según haya personal disponible dentro de la cuadrilla. Como ejemplo de consideraciones que el modelo incluye, las actividades pueden comenzar, detenerse y continuar luego de un período de tiempo. El trabajo concluye con resultados favorables encontrando para el case study analizado una planificación de tareas que implica una menor cantidad de días de trabajo y también una menor suma de costos con respecto a la planificación hallada por una alternativa propuesta en estudios previos (Zou, Fang, Huang, y Zhang, 2017) que utiliza un modelo MILP basado en fixed priority scheme and sequential

[4]Interval Variables - https://www.ibm.com/docs/en/icos/20.1.0?topic=concepts-interval-variables-in-cp-optimizer

[5]https://www.minizinc.org/doc-2.3.0/en/lib-globals.html

scheduling strategy (FPS&SSS).

La integración de CP en la disciplina de Investigación de Operaciones (Operations Research - OR) con el fin de obtener un enfoque de solución de problemas unificado es una tendencia que ha acaparado interés en las últimas décadas. Según expone *John N. Hooker* en el capítulo *XV* del libro *Handbook of Constraint Programming* (Rossi, van Beek, y Walsh, 2006), "Quizás siempre haya un lugar para los solvers especializados. Sin embargo, también se puede prever un futuro en el que los solvers de uso general basados en CP y OR de hoy en día evolucionen a sistemas integrados en los que ya no existe una distinción clara, ni necesidad de hacer una distinción entre componentes CP y OR."

Los problemas de asignación son justamente uno de los grupos de problemas que mejor se adaptan a las herramientas que CP ofrece. Este tipo de problemas tiene muchas variantes (allocation, schedulling, bin packing, etc) y cada una de ellas presenta dificultades diferentes y naturalezas en sus planteos que pueden ser mejor aprovechadas por CP o por MIP (Dimény y Koltai, 2023).

En las siguientes sub-secciones se listan algunos conceptos del paradigma de Constraint Programming que resultan de interés tenerlos presente para comprender su mención posteriormente en las demás secciones del informe.

2.5.1. Solvers

Los Solvers son potentes softwares encargados de llevar adelante la búsqueda de solucionas para los modelos escritos en lenguajes de CP. Llevan a cabo las resoluciones explorando sistemáticamente el espacio de búsqueda y las posibles asignaciones de valores (subdominios que las restricciones implican) sobre las variables del modelo. Utilizan técnicas de búsqueda heurística, inferencia lógica y poda para guiar la exploración y reducir el dominio de soluciones factibles. Estas reducciones también surgen como resultado de la "propagación de restricciones". La propagación de restricciones consiste en que si una restricción se satisface en una sección del problema, esto puede tener implicancias en otras secciones del mismo. Luego, el enfoque es aprovechar estas relaciones entre las restricciones para deducir información adicional (deducir nuevas restricciones a partir de las ya existentes) y con ello reducir el número de posibles valores en los dominios de otras variables.

Algunos solvers son gratuitos y de código abierto, mientras que otros son comerciales y requieren una licencia paga. Dentro de los pagos, algunos ejemplos son Gurobi o CPLEX. En este caso, se ha optado por un solver de licencia gratuita, teniendo en cuenta también la posibilidad de un futuro uso por parte de las cooperativas que cada vez más suelen optar por asignar sus viviendas con MTAV. Particularmente, las pruebas y análisis se han ejecutado utilizando Gecode [6]. Gecode es un solver open source, altamente portable ya que se encuentra implementado en C++ y es el solver por defecto incluido en el instalable de MiniZinc. Si bien ha sabido demostrar su capacidad y eficiencia en años anteriores ganando competencias como los MiniZinc Challenges, hoy día no habría una distinción o ventaja clara con respecto a otra opción como lo es el solver CP-SAT de Google OR-Tools [7] que también es un gran ganador de concursos en los últimos años y también es open source.

2.5.2. Propagation Strength

En Constraint Programming, la búsqueda dentro del espacio de soluciones puede ser guiada o configurada estableciendo reglas para la toma de decisiones. Estas reglas se especifican dentro del modelo mediante lo

[6] Gecode - https://www.gecode.org/

[7] CP-SAT - https://developers.google.com/optimization/cp/cp_solver

que se conoce como Anotaciones o Annotations [8], que luego el solver considera a la hora de iterar con una nueva ramificación de combinaciones de valores para el conjunto de variables del problema.

Existe una anotación que corresponde al modo en que la propagación de restricciones mencionada anteriormente se aplica. La anotación especifica la fuerza de propagación de una restricción en relación a los límites y el dominio de las variables. Las opciones son "Bounds" (límites) y "Domain" (dominio). La diferencia principal entre estas dos radica en la forma en que se enfoca la propagación de restricciones. "Bound" se enfoca en los límites específicos de las variables, mientras que "Domain" se enfoca en todo el rango de valores dentro del dominio de las variables. De esta forma, mientras que "Domain" propaga una restricción a cada valor dentro del dominio de una variable para validarlo o eliminarlo, "Bounds" solo verificará si a partir de dicha restricción, los límites del dominio de la variable pueden ser reducidos.

La elección de una opción u otra dependerá fuertemente del problema específico. En general, se puede afirmar que la opción "Bounds" tiende a ser más ligera de aplicar que la opción "Domain" en términos de procesamiento y consumo de recursos. Pero esto no implica que una opción sea siempre más eficiente que la otra para abordar la búsqueda de soluciones, por lo que es recomendable realizar pruebas para evaluar el rendimiento en cada caso.

2.5.3. Selección de Variable

Una de las principales anotaciones es la Selección de Variables; se refiere a la capacidad de establecer el orden en el que las variables del problema se seleccionan para luego modificar su valor actual y así iterar en una nueva ramificación del espacio solución. Definir un criterio para priorizar variables de decisión sobre las cuales progresar genera diferencias en cómo se cubre el árbol de soluciones, por lo que dependiendo el problema a abordar, un criterio puede ser fuertemente preferible en comparación a otros. Hay diferentes estrategias de selección de variables que se pueden utilizar, algunas de ellas son:

- **First_Fail**
  Selecciona primero la variable con el dominio más restrictivo, es decir, aquella que tiene menos posibles valores. Esta estrategia tiende a reducir el espacio de búsqueda más rápidamente, ya que se abordan las restricciones más fuertes en etapas tempranas de la búsqueda.
- **Anti_First_Fail**
  El opuesto a la anterior. Selecciona la variable con el dominio menos restrictivo, es decir, la que tenga más posibles valores.
- **Largest / Smallest**
  Selecciona aquella variable que tenga el valor más - alto / bajo - en su dominio.
- **Occurrence**
  Selecciona la variable que esté implicada en más cantidad de restricciones.

2.5.4. Selección de Valor

Una vez seleccionada una variable a la cual asignarle un próximo valor, de igual manera es posible también determinar la estrategia a seguir para tomar valores dentro del dominio específico de la variable. La anotación Selección de Valor o Value Selections, tiene una variedad de opciones, dentro de las más clásicas o intuitivas se puede encontrar la selección del valor más alto, o más bajo, o promedio, o simplemente aleatorio. A su vez otra capacidad es poder optar por una selección aleatoria pero solicitando que se excluya el valor más alto, bajo, o promedio.

Nuevamente, según el problema a resolver, esta anotación también ofrece un impacto claro en la eficiencia y las soluciones encontradas por parte de los solvers. Cuando no es requerido encontrar todas las soluciones sino únicamente satisfacer el problema, es donde más puede afectar.

[8] Annotations - https://www.minizinc.org/doc-2.3.0/en/lib-annotations.html

2.6. Herramientas

2.6.1. GLPK

GLPK (GNU Linear Programming Kit) [9] es un paquete de software open-source desarrollado para llevar a cabo la resolución de problemas de programación lineal de gran escala, problemas de programación entera mixta y problemas de programación matemática relacionados. Con ese enfoque, GLPK no soporta la resolución de modelos escritos con Constraint Programming.

El software GLPK es un conjunto de funciones construidas en ANSI C y organizadas en forma de librería. Puede utilizarse para la resolución directa de problemas mediante el intérprete de línea de comandos glpsol, intérprete que se usa en este trabajo para ejecutar los modelos actuales de MTAV.

Dentro de los principales componentes del paquete GLPK se encuentran motores de búsqueda con los siguientes métodos:

- primal and dual simplex
- primal-dual interior-point
- branch-and-cut

Una enorme ventaja de esta herramienta es ser Cross-Platform. GLPK es ejecutable en la mayoría de sistemas operativos, siendo los principales Linux, Windows, y macOS. Esto a su vez es un punto clave en la aplicación de MTAV para no generar dependencias extras en cuanto al ambiente de ejecución.

2.6.2. Minizinc

MiniZinc [10] es un lenguaje de modelado y un sistema de resolución de problemas de programación basada en restricciones. Con él, es posible describir problemas en términos de variables, restricciones y objetivos. El lenguaje es de alto nivel y sencillo de comenzar a utilizar, lo que facilita la formulación de modelos. Permite expresar restricciones lógicas, aritméticas, de dominio y varios otros tipos de parametrizaciones.

MiniZinc incluye un traductor que convierte los modelos a un formato comprensible para diferentes solucionadores de CP, como lo son Gecode, Chuffed, Gurobi o CPLEX, entre otros. Además de lo anterior, MiniZinc cuenta con un IDE para el desarrollo de soluciones y la ejecución de las mismas, por esto es considerado no solo un lenguaje sino un sistema.

Este es el framework de trabajo escogido para el desarrollo y ejecución de los modelos implementados en CP durante este trabajo. La motivación para esta selección surgió a partir de las características mencionadas previamente sobre MiniZinc en conjunto con el hecho de que es una herramienta que es activamente mantenida (el último release al momento figura con fecha 1 Febrero de 2024); cuenta con una buena documentación online [11], una amplia comunidad activa a través de distintos foros o portales y por último la particularidad de que es una herramienta gratuita y permite la utilización de la misma no solo para motivos de investigación sino también para usos en casos reales si se quisiera.

Otro indicio de interés para optar por MiniZinc es el compromiso que la organización asociada al framework tiene con respecto al área de estudio de la resolución de problemas con Constraint Programming: Desde el año 2008, cada año organizan un reconocido desafío (The MiniZinc Challenge [12]) con el objetivo de poner a prueba una variedad de distintos solvers y poder estudiar para el problema concreto del challenge, mediante benchmarks, cuales solvers se desenvuelven mejor.

---

[9] glpk - https://www.gnu.org/software/glpk/

[10] Minizinc - https://www.minizinc.org/

[11] Minizinc Handbook - https://www.minizinc.org/resources.html/

[12] The Minizinc Challenge - https://www.minizinc.org/challenge/

Capítulo 3

Implementación y resultados

En este capítulo se presentan distintos análisis y comparativas entre ambos enfoques disponibles siendo estos los ya mencionados, Programación Matemática y CP. En primera instancia, la comparación se realiza en términos de tiempo de cómputo, para ello se listan benchmarks de las ejecuciones de ambas soluciones sobre los mismos casos de prueba. Posteriormente se muestran los resultados y algunas de las primeras conclusiones obtenidas luego de estudiar un concepto de equidad sobre las posibles asignaciones disponibles que cada caso tiene en su dominio.

Para obtener una mejor visualización de estas opciones y lo que éstas ofrecen, se complementa con la construcción de un Frente de Pareto entre los dos objetivos presentes: Satisfacción Global y Equidad. "El Frente de Pareto es un conjunto de soluciones no dominadas, que se eligen como óptimas si ningún objetivo puede mejorarse sin sacrificar al menos otro objetivo. Por otro lado, se dice que una solución x* está dominada por otra solución x si, y sólo si, x es igual o mejor que x* con respecto a cada uno de los objetivos."(Mehdi Khosrow-Pour, 2017)

3.1. Modelos para asignación de viviendas usando CP

Considerando el interés de investigar una alternativa para la resolución de los problemas de asignaciones donde la representación del problema sea relativamente sencilla de adaptar a nuevos escenarios, se propone trasladar el modelo conceptual de MTAV actual al entorno de CP. Este enfoque ofrece posibilidades variadas tal como se ha mencionado previamente en el Capítulo 2 al introducir el paradigma alternativo elegido. En este lenguaje es posible transcribir la misma solución actualmente definida para MTAV a la vez que permite también incorporar nuevos aspectos como el investigado posteriormente con respecto a la medida de equidad que los resultados contienen.

Se mantiene el mismo esquema de búsqueda en dos etapas, donde la primer etapa es la obtención de la mejor cota Z para las satisfacciones individuales y la segunda etapa el hallazgo de la mejor solución colectiva. De igual manera, las restricciones y función objetivo se conservan. El modelo se ajusta para aprovechar las características de Constraint Programming, por tanto, a diferencia de la instancia matemática donde la estructura de variables de decisión se define con una matriz binaria asociando viviendas con cooperativistas, en este enfoque la estructura se puede implementar de distintas formas. Para este estudio, inicialmente se optó por experimentar con dos versiones por separado: un vector de prioridades asignadas y un vector de viviendas asignadas.

Vector de Prioridades

A continuación se presentan los modelos para la búsqueda de la mejor cota Z y de la mejor satisfacción colectiva utilizando como representación para las variables de decisión un vector de *prioridades* asignadas.

Una breve explicación se encuentra seguido de los mismos.

Modelo Intermedio - Búsqueda de la cota Z

```
include "alldifferent.mzn";

int: n;      % numero de Nucleos %

set of int: PREFS = 1..n;
set of int: VIVIENDAS = 1..n;
set of int: NUCLEOS = 1..n;

array[NUCLEOS,VIVIENDAS] of int: p; % prioridades %

array[NUCLEOS] of var PREFS: p_asignadas; % prioridad para cada nucleo %
array[NUCLEOS] of var VIVIENDAS: v_asignadas; % vivienda de cada nucleo %


% Asignaciones all different %
constraint forall(f in NUCLEOS, x in VIVIENDAS)
          (if p_asignadas[f] == p[f,x] then
            v_asignadas[f] = x
           else
            true
          endif);

constraint alldifferent(v_asignadas);


var int: z = max(p_asignadas);

solve :: int_search(p_asignadas, varselect, valselect, complete)
    minimize z;
```

Listing 3.1 Codigo Fuente - Minizinc - Vector de Prioridades - Cota Z

Modelo Final - Búsqueda de Satisfacción Global

```
include "alldifferent.mzn";

int: n;      % nro de Nucleos %
int: z;      % cota superior de Prioridad %

set of int: PREFS = 1..n;
set of int: VIVIENDAS = 1..n;
set of int: NUCLEOS = 1..n;

array[NUCLEOS,VIVIENDAS] of int: p;  % prioridades %

```

```
array[NUCLEOS] of var PREFS: p_asignadas; % prioridad para cada nucleo %
array[NUCLEOS] of var PREFS: v_asignadas; % vivienda de cada nucleo %

% Asignaciones all different %
constraint forall(f in NUCLEOS, x in PREFS)
        (if p_asignadas[f] == p[f,x] then
            v_asignadas[f] = x
         else
            true
         endif);

constraint alldifferent(v_asignadas);
constraint z >= max(p_asignadas);

var int: satisf = sum(p_asignadas);

solve :: int_search(p_asignadas, varselect, valselect, complete)
    minimize satisf;
```

Listing 3.2 Codigo Fuente - Minizinc - Vector de Prioridades - Satisfaccion Global

En esta primera versión, siendo los datos de entrada (al igual que en el modelo matemático) una matriz con las prioridades de los cooperativistas, la estructura de variables de decisión es un vector de prioridades **p_asignadas** que contiene la prioridad seleccionada para cada familia. De esta manera es sencillo definir la función objetivo como $\sum_{i \in N} \textbf{p\_asignadas}(i)$. Sin embargo, donde no resulta práctico este planteo es a la hora de verificar la biyectividad de las asignaciones, ya que para garantizar que una misma vivienda no esté asignada más de una vez, es necesario recorrer la matriz de prioridades para cada prioridad asignada en busca de la columna correspondiente. Con el fin de retener la información anterior para luego aplicar la restricción necesaria, se utiliza una estructura auxiliar **v_asignadas** que guarda la vivienda asignada a cada cooperativista. Ahora simplemente es necesario verificar que los valores no se repitan en dicha estructura auxiliar, construida en cada iteración a partir de las decisiones tomadas sobre **p_asignadas**. Esta versión tiene la desventaja de que durante la ejecución, en el peor caso la matriz es recorrida por completo con cada combinación de prioridades escogida durante la búsqueda y esto conlleva un costo computacional significante.

Vector de Viviendas

Nuevamente se encuentran, en los Listings 3.3 y 3.4 los modelos para ambas etapas de la solución aunque con la ligera modificación de definir las variables como un vector de *viviendas* asignadas. Tras los modelos, se incluye una breve explicación.

Modelo Intermedio - Búsqueda de la cota Z

```
include "alldifferent.mzn";

int: n;     % nro de Nucleos %

set of int: VIVIENDAS = 1..n;
set of int: NUCLEOS = 1..n;
```

```
7
8  array[NUCLEOS,VIVIENDAS] of int: p;  % prioridades %
9
10 array[NUCLEOS] of var VIVIENDAS: v_asignadas; % asignaciones %
11
12 % Asignaciones all different %
13 constraint alldifferent(v_asignadas);
14
15 % Peor prioridad asignada %
16 var int: z = max(f in NUCLEOS)( p[f,v_asignadas[f]] );
17
18 solve :: int_search(v_asignadas, varselect, valselect, complete)
19     minimize z;
```

Listing 3.3 Codigo Fuente - Minizinc - Vector de Viviendas - Cota Z

Modelo Final - Búsqueda de Satisfacción Global

```
1  include "alldifferent.mzn";
2
3  int: n;   % nro de Nucleos %
4  int: z;     % peor posible Asignacion %
5
6  set of int: VIVIENDAS = 1..n;
7  set of int: NUCLEOS = 1..n;
8
9  array[NUCLEOS,VIVIENDAS] of int: p;  % prioridades %
10
11 array[NUCLEOS] of var VIVIENDAS: v_asignadas;  % vivienda de cada nucleo %
12
13 % Asignaciones all different.
14 constraint alldifferent(v_asignadas);
15
16 % Max preferencia asignada es a lo sumo valor z.
17 constraint forall(f in NUCLEOS)( p[f,v_asignadas[f]] <= z );
18
19 var int: satisf = sum(f in NUCLEOS)( p[f,v_asignadas[f]] );
20
21 solve :: int_search(v_asignadas, varselect, valselect, complete)
22     minimize satisf;
```

Listing 3.4 Codigo Fuente - Minizinc - Vector de Viviendas - Satisfaccion Global

Para resolver el problema de la estructura auxiliar de la primera versión, se escoge ahora una estructura de datos similar donde las variables de decisión no representan las prioridades sino directamente las viviendas asignadas, es decir, **v_asignadas**. De esta manera, no es necesario iterar en cada fila de la matriz de prioridades para asegurar una asignación biyectiva y aún para el cálculo de la función objetivo, los accesos a

la matriz para obtener las prioridades correspondientes son de orden 1, ya que el vector de variables tiene la información de fila y columna para esto. Sin embargo, esta versión tiene un problema diferente, de carácter más genérico, y es que al iterar sobre números de viviendas en vez de prioridades, las estrategias para la selección del próximo valor dada una variable escogida para modificar, se vuelven inservibles. No es posible discernir cuál vivienda es mejor asignar para cada caso ya que para calcular su impacto en la satisfacción global, primero se necesita acceder a la prioridad asociada. Resulta entonces, para esta versión, que el barrido del espacio de soluciones no sigue ningún lineamiento.

Vector de Prioridades - V2

Es claro que ambas versiones previas tienen sus ventajas pero también sus puntos débiles. De esas dos opciones surge entonces una tercera propuesta que consta en:

- Volver a las variables de decisión representando prioridades. Esto permite nuevamente disponer de la posibilidad de aplicar distintas estrategias de selección de variable y valor.
- Añadir una nueva estructura de datos de entrada. Para evitar la iteración constante sobre la matriz de prioridades inicial $P_{(f,v)}$, se agrega una segunda versión de la misma información pero reordenada de forma que esta nueva matriz $V'_{(f,p)}$ retorna la vivienda correspondiente para un cooperativista dada la prioridad.

Con este nuevo dato, procesado previo a la ejecución de la búsqueda de asignaciones, el costo computacional de la ejecución es menor.

Modelo Intermedio - Búsqueda de la cota Z

```
include "alldifferent.mzn";

int: n;   % nro de Nucleos %

set of int: PREFS = 1..n;
set of int: VIVIENDAS = 1..n;
set of int: NUCLEOS = 1..n;

array[NUCLEOS,VIVIENDAS] of int: p; % prioridades %
array[NUCLEOS,PREFS] of int: ov; % viviendas ordenadas por prioridad %

array[NUCLEOS] of var PREFS: p_asignadas; % de nucleo - prioridad %
array[NUCLEOS] of var VIVIENDAS: v_asignadas; % de nucleo - vivienda %


% Asignaciones all different
constraint forall(f in NUCLEOS) (v_asignadas[f] = ov[f,p_asignadas[f]]);
constraint alldifferent(v_asignadas);

var int: z = max(p_asignadas);

solve :: int_search(p_asignadas, varselect, valselect, complete)
    minimize z;
```

Listing 3.5 Minizinc - Vector de Prioridades (V2) - Cota Z

Modelo Final - Búsqueda de Satisfacción Global

```
include "alldifferent.mzn";

int: n;    % nro de Nucleos %
int: z;

set of int: PREFS = 1..n;
set of int: VIVIENDAS = 1..n;
set of int: NUCLEOS = 1..n;

array[NUCLEOS,VIVIENDAS] of int: p;  % prioridades %
array[NUCLEOS,PREFS] of int: ov;  % viviendas ordenadas por prioridad %

array[NUCLEOS] of var PREFS: p_asignadas; % de nucleo - su prioridad %
array[NUCLEOS] of var VIVIENDAS: v_asignadas; % de nucleo - su vivienda %


% Asignaciones all different %
constraint forall(f in NUCLEOS) (v_asignadas[f] = ov[f,p_asignadas[f]]);
constraint alldifferent(v_asignadas);

% Max preferencia asignada es a lo sumo z %
constraint z >= max(p_asignadas);

var int: satisf = sum(p_asignadas);

solve :: int_search(p_asignadas, varselect, valselect, complete)
    minimize satisf;
```

Listing 3.6 Minizinc - Vector de Prioridades (V2) - Satisf. Global

3.2. Propuesta de nueva medida de equidad: Desviación Estándar

La desviación estándar es una herramienta estadística muy presente en el análisis de datos. Proporciona información sobre la dispersión de un conjunto de datos, es decir, mide cuánto se desvían individualmente los valores de un conjunto con respecto a la media de ese mismo conjunto. En el contexto del beneficio individual dentro de un grupo de personas, la desviación estándar puede utilizarse como una medida de equidad para evaluar cuán uniformemente distribuidos están los beneficios. Si se nota como x_i la satisfacción de cada cooperativista, y $\bar{x}$ la satisfacción promedio, entonces la desviación estándar de la satisfacción se puede calcular con esta fórmula:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

Luego, una vez calculada la desviación estándar es posible encontrar dos situaciones claras y opuestas según el valor obtenido. Si la desviación estándar es alta, puede sugerir que algunos individuos están muy satisfechos mientras que otros están muy insatisfechos. Esto podría interpretarse como una señal de desigualdad en la distribución de la satisfacción. Por el lado contrario, una desviación estándar baja indica que las puntuaciones de satisfacción individual están cerca del promedio, lo que se puede interpretarse como una distribución más equitativa.

Es importante señalar que la desviación estándar es solo una de las muchas medidas que se pueden utilizar para evaluar la equidad o la distribución de valores dentro de las soluciones obtenidas. Otras medidas, como el rango intercuartílico o el coeficiente de asimetría, también pueden proporcionar información adicional sobre la forma y la dispersión de la distribución de datos. Para el caso puntual de la realidad presente en este trabajo la desviación estándar es suficiente como medida y sirve también como objetivo concreto sobre el cual buscar un óptimo.

La incorporación de un análisis de la equidad, buscando lograr el mejor valor de ésta, puede definirse como una extensión del mecanismo de MTAV que incluye una tercera etapa. La nueva etapa entonces tomaría el valor de satisfacción global junto a la cota superior Z y resolvería el problema de hallar la asignación con menor desviación estándar que cumpla con los óptimos de las etapas previas. Nuevamente se puede visualizar la solución completa empleada como una búsqueda lexicográfica de tres objetivos distintos, cada uno cubriendo un aspecto diferente que garantiza que la asignación encontrada es la mejor posible para la cooperativa como grupo.

3.3. Modelo de CP incluyendo la nueva medida de equidad

En este otro modelo, que toma como base y extiende el modelo con vector de prioridades en su segunda versión (Listing 3.6), se incorpora el valor de la satisfacción global como dato de entrada para asegurar que toda distribución de viviendas cumpla con dicha condición al igual que con el valor de cota superior Z. Se puede observar también que se espera un nuevo vector de datos (**warm_p**) que pretende utilizarse como conjunto de valores inicial para las variables de selección, tomando ventaja de las búsquedas previamente hechas. Sin embargo, el solver utilizado Gecode aún no soporta esta funcionalidad aunque el lenguaje MiniZinc sí permitiera especificarla, concluyendo en la omisión de dichos valores durante la ejecución de este modelo. La nueva restricción impide que la suma de prioridades varíe con respecto a la satisfacción especificada en los parámetros y luego la función objetivo en este caso es la desviación estándar la cual se intenta minimizar (o maximizar según el análisis que se deseee realizar).

Modelo Equidad - Búsqueda de mayor/menor desviación estándar

```
include "alldifferent.mzn";

int: n;      % nro de Nucleos %
int: z;        % cota maxima de Prioridad %
int: sgo;      % Satisf Global %

set of int: PREFS = 1..n;
set of int: VIVIENDAS = 1..n;
set of int: NUCLEOS = 1..n;

array[NUCLEOS] of PREFS: warm_ps;  % warm prioridades %
array[NUCLEOS,VIVIENDAS] of int: p;  % prioridades %
array[NUCLEOS,PREFS] of int: ov;  % viviendas ordenadas por prioridad %

```

```
15 array[NUCLEOS] of var PREFS: p_asignadas; % de nucleo - su prioridad %
16 array[NUCLEOS] of var VIVIENDAS: v_asignadas; % de nucleo - su vivienda %
17
18
19 var float: x_mean = sum(p_asignadas) / n;
20 var float: std_dev =
21   sqrt(
22     sum(e in NUCLEOS)(
23         p_asignadas[e]*p_asignadas[e]
24         + x_mean*x_mean
25         - 2*p_asignadas[e]*x_mean
26     ) / n
27   );
28
29
30 % Satisfaccion Global fija %
31 constraint sgo = sum(p_asignadas);
32
33 % Cota maxima de prioridad
34 constraint z >= max(p_asignadas);
35
36 % Asignaciones all different
37 constraint forall(f in NUCLEOS) (v_asignadas[f] = ov[f,p_asignadas[f]]);
38 constraint alldifferent(v_asignadas);
39
40 solve :: warm_start(p_asignadas, warm_ps)
41       :: int_search(p_asignadas, varselect, valselect, complete)
42       [maximize / minimize] std_dev;
```

Listing 3.7 Minizinc - Vector de Prioridades (V2) - Equidad

3.4. Casos de prueba

Como conjunto de datos para las pruebas de los modelos descriptos se optó por usar casos reales de cooperativas que han hecho sus asignaciones con MTAV. En lugar de generar sorteos auxiliares, se consideró como una mejor muestra a las prioridades reales de cooperativistas, ya que de esta manera es posible asegurar una mayor aproximación en cuanto a cómo las prioridades grupales suelen distribuirse y relacionarse relativamente entre sí. El equipo de MTAV cuenta con un amplio historial de casos ya resueltos con éxito y estos datos fueron puestos a disposición para la realización de la tesis. A continuación se encuentran listados los casos que se seleccionaron y en el Anexo se puede acceder a la matriz de prioridades asociada a cada uno.

- [Caso A] Asign. de 5 viviendas.
- [Caso B] Asign. de 20 viviendas.
- [Caso C] Asign. de 22 viviendas.
- [Caso D] Asign. de 11 viviendas.
- [Caso E] Asign. de 22 viviendas.
- [Caso F] Asign. de 18 viviendas.
- [Caso G] Asign. de 19 viviendas.
- [Caso H] Asign. de 4 viviendas.
- [Caso I] Asign. de 10 viviendas.
- [Caso J] Asign. de 14 viviendas.
- [Caso K] Asign. de 4 viviendas.
- [Caso L] Asign. de 13 viviendas.
- [Caso M] Asign. de 29 viviendas.
- [Caso N] Asign. de 29 viviendas.

Se puede apreciar entre los casos escogidos que los mismos varían en cantidad de viviendas. Una misma cooperativa muchas veces realiza múltiples sorteos. Esto es debido a que se componen de viviendas con distintas prestaciones, donde la mayor diferencia suele encontrarse en la cantidad de dormitorios, por lo tanto se realiza un sorteo por cada categoría de vivienda. Generalmente (al menos desde la puesta a disposición pública de MTAV y hasta la fecha) más del 90 % de las asignaciones se encuentran entre estas cantidades de viviendas a la hora de realizar el sorteo, promediando entre 15 y 25 viviendas.

3.5. Desempeño de soluciones

En el capítulo anterior se introdujeron los conceptos de estrategias de selección de variables y selección de valores asociados al enfoque CP. Estas estrategias cumplen un rol importante al momento de cubrir el espacio de soluciones y su implicancia en las diferencias de desempeño varían según el problema a resolver.

El problema de asignaciones de este trabajo tiene la particularidad que el dominio de valores para las variables es el mismo para todas, y de igual forma ocurre con las restricciones, donde las mismas afectan de igual manera a todas las variables. Sin embargo, se ve en los listados de esta sección que aún en dicho contexto las estrategias son fundamentales y es deseable analizarlas.

Las principales estrategias disponibles para el solver utilizado en las ejecuciones son las siguientes:

Selección de Variables

- First Fail (variable con el dominio más chico)
- Anti First Fail (variable con el dominio más grande)
- Largest (variable con el valor más alto en el dominio)
- Smallest (variable con el valor más bajo en el dominio)
- Input Order
- Max Regret (variable con la diferencia más grande entre los dos valores más chicos de su dominio)

Selección de Valores

- Indomain (valores en orden ascendente)
- Indomain Max (valor más alto en el dominio)
- Indomain Median (valor medio en el dominio)
- Indomain Min (valor más bajo en el dominio)
- Indomain Random (valor aleatorio)
- Indomain Split (bisectar el dominio, excluyendo la mitad superior primero)
- Indomain Reverse Split (bisectar el dominio, excluyendo la mitad inferior primero)
- Indomain Split Random (bisectar el dominio, excluyendo aleatoriamente una mitad primero)

La presente sección se divide en dos subsecciones. La inicial contiene una comparación de benchmarks entre ambos enfoques de resolución (MIP y CP), fijando siempre, para el caso de CP, una misma estrategia de búsqueda. La posterior subsección expone una prueba más exhaustiva de las estrategias de búsqueda haciendo especialmente foco en aquellos casos de prueba de mayor magnitud para los cuales la primera etapa dejó un margen de mejora más amplio en términos de tiempos de ejecución o resultados.
Los modelos de CP utilizados en ambas etapas de pruebas son los vistos en los listados 3.3, 3.4, 3.5 y 3.6 y todas las ejecuciones de los mismos fueron limitadas a tiempos de máximo 75 segundos, tras lo cual si no se logra confirmar un óptimo, se retorna el mejor valor hallado al momento.

3.5.1. Benchmarks Comparativos - Primera Etapa

En la Tabla 3.1 se listan los tiempos necesarios por los distintos enfoques para hallar tanto el valor Z de la primer etapa como las asignaciones de la etapa posterior. Esta primera iteración se ejecutó con las siguientes estrategias de selección: *input_order* para la selección de variable y *random* para la selección de valor. Los valores en negrita indican que el óptimo fue alcanzado, ✗ indica que dentro del tiempo establecido ninguna solución factible fue hallada y ~ 0 que el tiempo requerido fue menor a una centésima de segundo. El solver GLPK para el modelo MIP encuentra en todos los casos el valor óptimo, por lo que es el valor de referencia para los otros resultados. Para CP, en caso de no lograr el valor óptimo dentro de los 75 segundos, el resultado listado es el mejor hallado y el tiempo que llevó encontrar dicha solución.

Caso	Enfoque	Z	Tiempo	Sol	Tiempo
A - x5	glpk	**4**	~ 0	**9**	~ 0
A - x5	vviv	**4**	0,33	**9**	0,57
A - x5	vpri	**4**	0,13	**9**	0,11
B - x20	glpk	**11**	~ 0	**73**	~ 0
B - x20	vviv	**11**	0,35	82	59,03
B - x20	vpri	**11**	0,11	83	71,65
C - x22	glpk	**7**	~ 0	**80**	~ 0
C - x22	vviv	**7**	0,41	**80**	26,96
C - x22	vpri	**7**	4,81	84	65,28

D - x11	glpk	**6**	~ 0	**24**	~ 0
D - x11	vviv	**6**	0,36	**24**	0,44
D - x11	vpri	**6**	0,16	**24**	0,12
E - x22	glpk	**5**	~ 0	**45**	~ 0
E - x22	vviv	**5**	0,35	**45**	1,87
E - x22	vpri	**5**	0,17	**45**	3,20
F - x18	glpk	**14**	~ 0	**62**	~ 0
F - x18	vviv	15	19,15	✗	75
F - x18	vpri	15	41,31	92	73,49
G - x19	glpk	7	~ 0	67	~ 0
G - x19	vviv	**7**	2,84	**67**	18,59
G - x19	vpri	**7**	0,22	**67**	55,18
H - x4	glpk	**3**	~ 0	**8**	~ 0
H - x4	vviv	**3**	0,35	**8**	0,50
H - x4	vpri	**3**	0,16	**8**	0,10
I - x10	glpk	**5**	~ 0	**31**	~ 0
I - x10	vviv	**5**	0,38	**31**	0,50
I - x10	vpri	**5**	0,12	**31**	0,10
J - x14	glpk	**8**	~ 0	**45**	~ 0
J - x14	vviv	**8**	0,36	**45**	0,47
J - x14	vpri	**8**	5,14	**45**	3,81
K - x4	glpk	**1**	~ 0	**4**	~ 0
K - x4	vviv	**1**	0,40	**4**	0,51
K - x4	vpri	**1**	0,10	**4**	0,10
L - x13	glpk	**5**	~ 0	**43**	~ 0
L - x13	vviv	**5**	0,39	**43**	0,53
L - x13	vpri	**5**	0,11	**43**	0,16
M - x29	glpk	14	~ 0	**159**	~ 0
M - x29	vviv	14	0,38	195	75
M - x29	vpri	14	9,37	215	28,09
N - x29	glpk	**8**	~ 0	**96**	~ 0
N - x29	vviv	**8**	0,48	121	17,57
N - x29	vpri	12	0,12	118	47,31
Caso	**Enfoque**	**Z**	**Tiempo**	**Sol**	**Tiempo**

Tabla 3.1: Resultados y benchmarks - I

Se puede observar a partir de los datos anteriores que las estrategias utilizadas - cuyo plan es simplemente probar valores aleatoriamente - funciona de forma exitosa para sorteos chicos y con valores de Z que sean relativamente bajos con respecto al tamaño del sorteo. En dichos casos la cantidad de combinaciones posibles es menor cualquiera sea el caso (sorteo chico o Z chico) y al no llevar a cabo cálculos para una selección más inteligente, los tiempos totales igual se mantienen en unos pocos segundos. Sin embargo, estas mismas estrategias tienen un desempeño desfavorable para sorteos grandes con más de 20 viviendas o para aquellos sorteos cuyo valor de cota superior Z no restringe significativamente el dominio de prioridades utilizables. Frente a estos casos, el enfoque no solamente tarda mucho más tiempo en encontrar su mejor solución sino que también esta tiende a ser alejada del óptimo para el problema.

3.5.2. Benchmarks Comparativos - Segunda Etapa

Dadas las diferencias anteriores con respecto al modelo de programación matemática (en las calidades de las soluciones retornadas) se presentan ahora valores de una subsecuente iteración de ejecuciones que intenta acortar la brecha combinando estrategias de búsqueda alternativas. Recordar que de los dos modelos de CP usados, aquí solamente será utilizado el que dispone las variables de decisión como un vector de prioridades (referenciado en las tablas como *vpri*) ya que son las prioridades de los usuarios las que están directamente ligadas a la función objetivo permitiendo aplicar estrategias más fácilmente a diferencia del modelo de viviendas.

A partir del listado a continuación, cada estrategia de selección de variable se ejecuta combinando con cada una de las estrategias de selección de valor.

Selección de Variables

- Input Order
- Largest
- Max Regret
- Smallest

Selección de Valores

- Indomain Median
- Indomain Split
- No fixed strategy

Las restantes estrategias de selección de variable (no incluidas en la lista anterior) fueron descartadas por trabajar sobre la cantidad de valores dentro de los dominios, debido a que para el caso de este problema de asignación, las prioridades se mapean a las viviendas y por tanto, al fijar una prioridad se está también fijando una vivienda la cual finalmente implica que los dominios de todas las variables (las prioridades individuales de los cooperativistas) pierdan el valor asociado a la vivienda estipulada. Lo anterior concluye en que todos los dominios comparten la misma cantidad de valores posibles restantes.

Las estrategias de selección de valor se escogieron en base a lo previamente observado en los resultados de sorteos realizados por las cooperativas. En los mismos, los valores de prioridad más frecuentes son aquellos que se encuentran inferiormente cercanos a la media del rango 1 a Z (cota superior de prioridad). Dicho de otra manera, la cantidad de prioridades fijadas, con su valor siendo Z o cercano a Z suele ser menor que aquellas con sus valores cercanos a 1 o en la media del rango. Por lo anterior se decide utilizar los valores medios en una primera iteración y luego utilizar la segmentación del dominio de modo que se usen los valores de 1 a Z/2 primero (que son más probables de ser correctos en una solución final óptima).

Una nota sobre la Tabla 3.2 de resultados es que la misma solo muestra los sorteos más complejos de resolver para evitar incluír casos que no son del todo interesantes o relevantes. El valor de Z configurado para cada caso es el óptimo hallado con GLPK.

Caso	**Enfoque / Estrategias**	**Valor Objetivo**	**Tiempo**
B - x20	glpk	**73**	~ 0
	input_order + median	83	49,18
	largest + median	85	23,76
	smallest + median	83	46,63
	max_regret + median	85	57,21
	input_order + split	74	3,85

	largest + split	✗	75,00
	smallest + split	74	1.48
	max_regret + split	**73**	2,87
	input_order + default	74	3,90
	largest + default	74	2,65
	smallest + default	**73**	10,47
	max_regret + default	**73**	2,18
C - x22	glpk	**80**	~ 0
	input_order + median	84	41,39
	largest + median	84	52,53
	smallest + median	84	21,46
	max_regret + median	84	54,67
	input_order + split	**80**	14,50
	largest + split	✗	75,00
	smallest + split	**80**	1,43
	max_regret + split	**80**	8,78
	input_order + default	**80**	14,58
	largest + default	**80**	11,67
	smallest + default	**80**	1,24
	max_regret + default	**80**	6.51
N - x29	glpk	**96**	~ 0
	input_order + median	125	19,89
	largest + median	127	73,48
	smallest + median	128	35,56
	max_regret + median	✗	75,00
	input_order + split	114	2,04
	largest + split	✗	75,00
	smallest + split	✗	75,00
	max_regret + split	114	4,37
	input_order + default	114	2,02
	largest + default	✗	75,00
	smallest + default	✗	75,00
	max_regret + default	113	35,14
F - x18	glpk	**62**	~ 0
	input_order + median	✗	75,00
	largest + median	✗	75,00
	smallest + median	✗	75,00
	max_regret + median	77	72,36
	input_order + split	✗	75,00
	largest + split	✗	75,00
	smallest + split	✗	75,00
	max_regret + split	**62**	1,57
	input_order + default	✗	75,00
	largest + default	✗	75,00
	smallest + default	✗	75,00
	max_regret + default	**62**	1,13
G - x19	glpk	**67**	~ 0
	input_order + median	**67**	69,13

	largest + median	**67**	1,76
	smallest + median	69	7,56
	max_regret + median	69	13,92
	input_order + split	**67**	7,27
	largest + split	✗	75,00
	smallest + split	**67**	3,65
	max_regret + split	**67**	9,54
	input_order + default	**67**	7,26
	largest + default	**67**	12,66
	smallest + default	**67**	1,24
	max_regret + default	**67**	5,33
M - x29	glpk	**159**	~ 0
	input_order + median	212	18,65
	largest + median	✗	75,00
	smallest + median	✗	75,00
	max_regret + median	222	31,09
	input_order + split	188	9,44
	largest + split	✗	75,00
	smallest + split	187	58,79
	max_regret + split	174	20,86
	input_order + default	188	9,61
	largest + default	197	60,32
	smallest + default	187	35,75
	max_regret + default	172	73,67
Caso	**Enfoque / Estrategias**	**Valor Objetivo**	**Tiempo**

Tabla 3.2: Resultados y benchmarks - II

Con estas nuevas iteraciones se puede observar que en términos generales, para este problema, existen mejores y peores estrategias tanto según el tiempo utilizado como también la solución devuelta por cada una. Se destaca que dicha diferencia de desempeño resulta de la combinación de ambos criterios - el de selección de variable y el de selección de valor - pues cambiando solamente uno de ambos, los resultados pueden ser completamente distintos.
Aunque no sea posible definir estrictamente una estrategia como la mejor debido a que en algunos casos eso no se cumple, sí es posible destacar que la combinación de estrategias de selección - max_regret & split - suele ser la que obtiene mejores resultados en todos los sorteos. Aún cuando otra combinación en casos particulares mejora los tiempos, los tiempos de la combinación mencionada no son muy diferentes.
El solver utilizado (Gecode) omite una estrategia si ésta no es soportada internamente, y procede a utilizar criterios y heurísticas escogidas libremente por el algoritmo dependiendo de las características del problema presente. En estos casos, dentro de los logs generados por el solver no se encuentra especificado si se ejecutó una estrategia en concreto de forma constante o si el plan es más complejo que al seleccionar una estrategia soportada. Sin embargo, los resultados de esta variante se asemejan bastante a los de la estrategia destacada previamente, siendo ligeramente mejores en la mayoría de los casos.
Finalmente, comparado a GLPK una vez más, ahora se consigue que las brechas no sean tan significativas entre ambos enfoques considerando que las satisfacciones globales obtenidas son iguales o similares. Aún en los sorteos más complejos donde las soluciones difieren, el tiempo límite utilizado puede perfectamente ser extendido si se requiere realizar una búsqueda más exhaustiva. Una ventaja que se presenta de utilizar Constraint Programming es la posibilidad de atacar problemas con funciones objetivo más complejas y variadas,

por ejemplo funciones cuadráticas tal como la desviación estándar que se escogió para analizar de alguna manera la equidad presesente en cada solución encontrada.

Los resultados y observaciones sobre este último punto es lo que se presenta en la siguiente sección.

3.6. Frente de Pareto sobre un concepto de Equidad

En la optimización multiobjetivo, el concepto de optimalidad de Pareto juega un papel fundamental al reflejar los compromisos entre objetivos conflictivos. Un estudio del Frente de Pareto actúa como una herramienta para visualizar y comprender dichos compromisos, presentando una perspectiva completa de las soluciones óptimas de Pareto dentro del dominio de los objetivos.

En esta sección, tomando algunos de los casos reales de asignaciones de viviendas en cooperativas, se presenta la exploración de soluciones óptimas de Pareto en base a dos conceptos (que a pesar de poder definirse de diferentes maneras dependiendo de cada realidad) se pueden considerar presentes en todo problema de asignación acompañado de prioridades sobre los objetos a asignar. Los conceptos a utilizar son la satisfacción global resultante para una solución o asignación dada y la equidad que ésta presenta, es decir, qué diferencias se hallan entre las satisfacciones individuales.

Para la realidad presente, la satisfacción global será la suma de las prioridades asignadas y por lo tanto, fácilmente se puede establecer la satisfacción individual como simplemente la preferencia asignada de cada uno. Recordar que se escoge la desviación estándar entre prioridades asignadas para marcar qué valor de "equidad" contiene cada solución.

Podemos formular entonces, algunas preguntas en base a lo planteado previamente:

- ¿Cómo varía este valor de equidad si se permitiera relajar el valor de satisfacción global?
- ¿Valdría la pena priorizar una mejor equidad en pos de una peor satisfacción global?
- ¿Es posible obtener distintas equidades dado un valor fijo de satisfacción global?

Para responder estas preguntas se ejecutaron múltiples iteraciones de la búsqueda de asignaciones utilizando rangos de valores de soluciones globales que comienzan en el valor óptimo de dicho objetivo para cada caso. El modelo utilizado para esta etapa es el presentado en la sección 3.3 que incluye la desviación estándar como función objetivo, y la estrategia de selección configurada es la considerada globalmente más eficaz a partir de los benchmarks de las secciones anteriores: *Max Regret* (variable_selection) y *Split* (value_selection). En cada iteración, se realizaron búsquedas de soluciones con el menor y mayor valor de equidad posibles intentando responder a la última pregunta listada. A continuación, se muestran gráficos para visualizar los resultados obtenidos de todas las ejecuciones junto a comentarios y conclusiones sobre los mismos. El conjunto de puntos resaltados en los gráficos (puntos marcados como hexágonos, de color verde) conforman los frentes de Pareto.

Una nota a considerar sobre las iteraciones es que todas ellas se realizaron con el parámetro de cota superior de prioridad (Z) hallado para cada caso al buscar los óptimos de satisfacción global. Además, las búsquedas de peor y mayor equidad en cada iteración se limitaron con timeouts de treinta minutos (lo anterior para contextualizar las ejecuciones de manera similar a las ejecuciones reales de asignaciones, donde al realizarlas en vivo, el tiempo límite no es infinito).

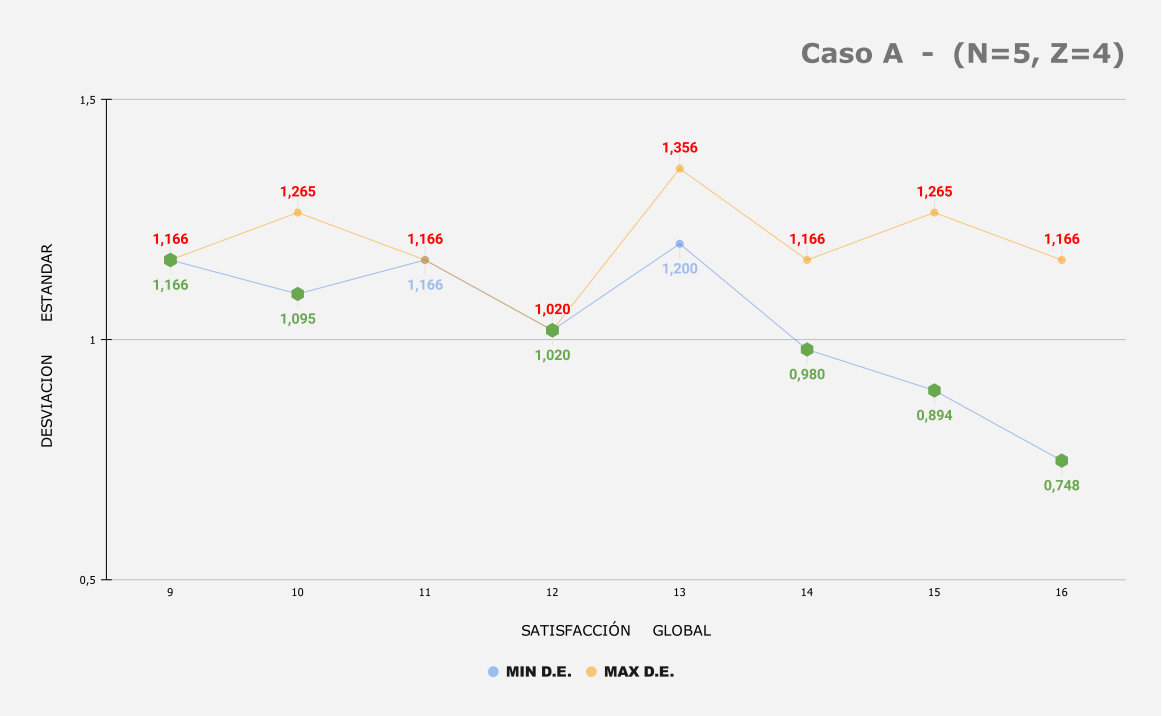


Figura 3.1: Frente de Pareto - [Caso A]

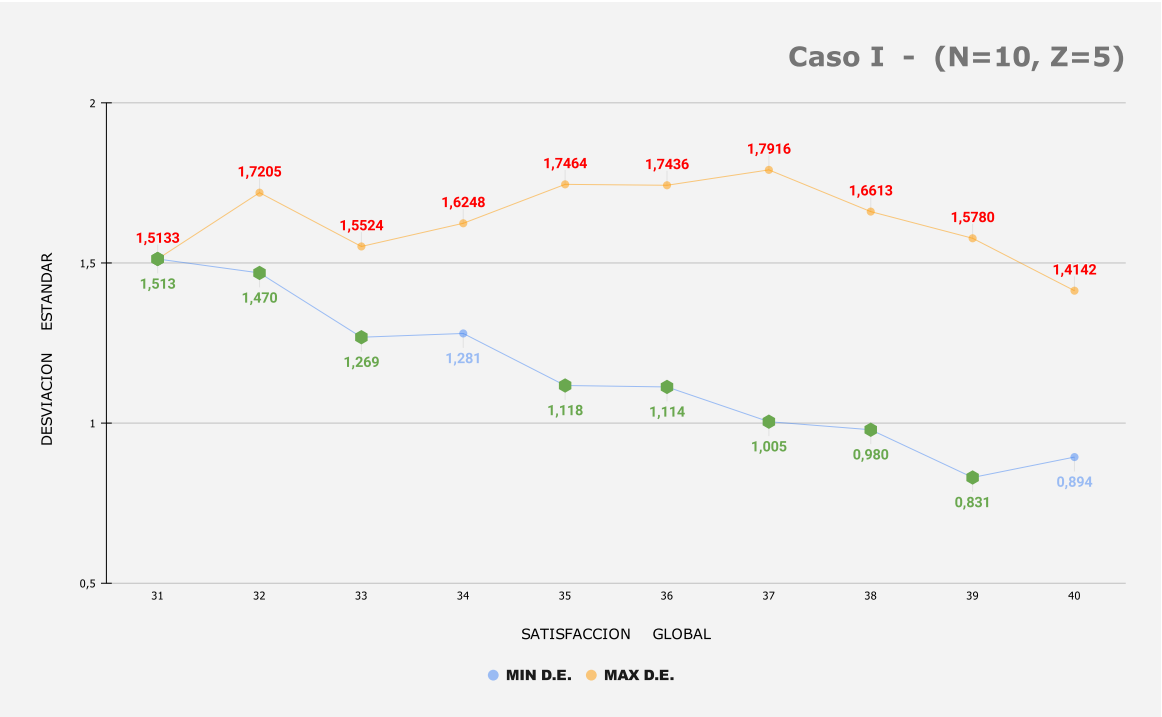


Figura 3.2: Frente de Pareto - [Caso I]

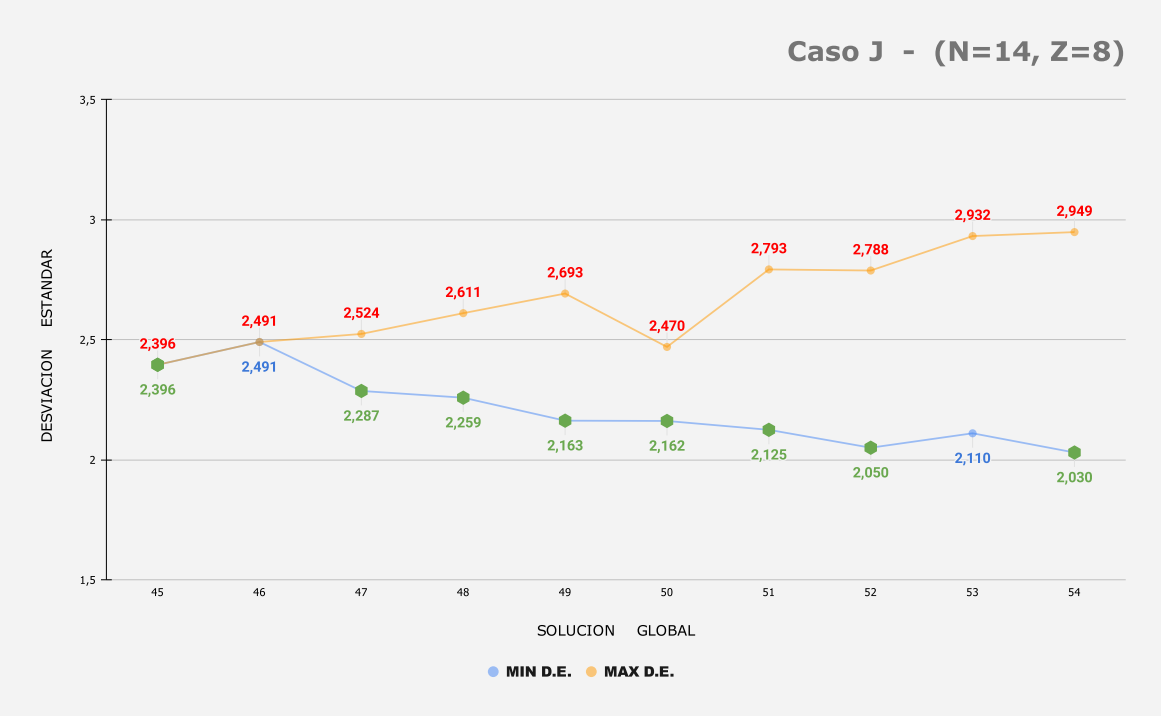


Figura 3.3: Frente de Pareto - [Caso J]

A partir de las Figuras 3.1, 3.2 y 3.3 es posible visualizar algunos puntos clave sobre la relación entre ambas métricas y responder la primer pregunta formulada. En primera instancia, se puede observar rápidamente, a partir del Frente de Pareto, que relajando la satisfacción global es posible obtener mejores valores de equidad. Esta es la tendencia general si se contempla completamente el eje horizontal de los gráficos, sin embargo, los valores de equidad oscilan y, para un valor dado de satisfacción global, la mejor equidad posible puede ser peor que la asociada a la solución global óptima de satisfacción global. Esto se aprecia en los tres casos A, I y J.

Por la naturaleza del problema, suele coincidir que en los valores extremos de la función objetivo, es decir la satisfacción global, el número de combinaciones en las asignaciones que resultan en dichos valores extremos sea mucho más reducida que en valores intermedios de satisfacción global. Por esta razón es que también se pueden esperar valores de equidad peores (una desviación estándar más amplia) en estos extremos mencionados.
Esta tendencia es observable en los resultados presentados a la vez que pueden visualizarse casos excepcionales en los que la equidad es aún peor que en el valor óptimo de satisfacción (motivo por el cual dichos puntos no son considerados dentro del Frente de Pareto).

Continuando con el análisis general de los resultados, y respondiendo a la tercer pregunta, se puede afirmar que para una misma satisfacción global en la mayoría de los casos es posible obtener diferentes asignaciones donde no todas necesariamente comparten la misma equidad. Este es un dato de gran relevancia para continuar con el desarrollo y la aplicación de MTAV como tecnología de asignación de viviendas ya que los algoritmos implementados en la herramienta actualmente en uso buscan la mejor asignación siguiendo criterios de optimizar las satisfacciones individuales y el promedio, pero no incorporan ninguna consideración respecto al concepto de equidad tal como definido en el trabajo presente.

La posibilidad de escoger una asignación dentro del conjunto que compone el Frente de Pareto es una interesante incorporación a evaluar para la herramienta MTAV, permitiendo a los cooperativistas optar, de alguna manera, cual criterio priorizar. Afirmar que siempre sería deseable priorizar la equidad sobre la satisfacción global es un tanto complejo, ya que las diferencias de equidad dentro del Frente de Pareto no siempre son realmente considerables a pesar de que la satisfacción global sufre una pérdida constante de una unidad en cada paso a lo largo del frente. Cuando la cantidad de viviendas N es menor y por lo tanto también lo es el valor óptimo de satisfacción global, resulta más difícil fundamentar la pérdida de unidades de satisfacción en favor de la equidad si la diferencia de ésta no es lo suficientemente notoria.

Por otro lado, la situación que resulta de mayor interés investigar es la siguiente: ¿Podría ocurrir, una vez obtenida la satisfacción global óptima, restringida por Z, que se deba analizar la equidad con el fin de garantizar que no se esté perdiendo la capacidad de mejorar dicha solución en términos de equidad?
En los gráficos 3.4, 3.5 y 3.6 se presenta alguno de los casos en donde esto sí ocurre y puede ser una gran mejora a considerar para futuros sorteos.

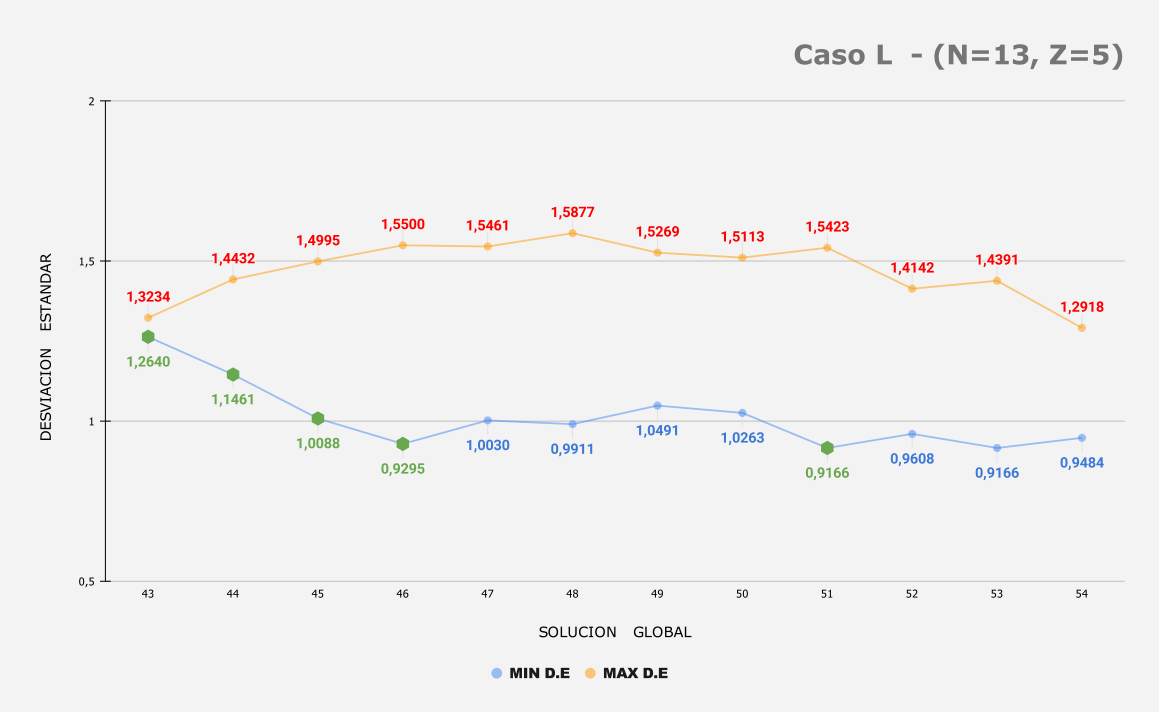


Figura 3.4: Frente de Pareto - [Caso L]

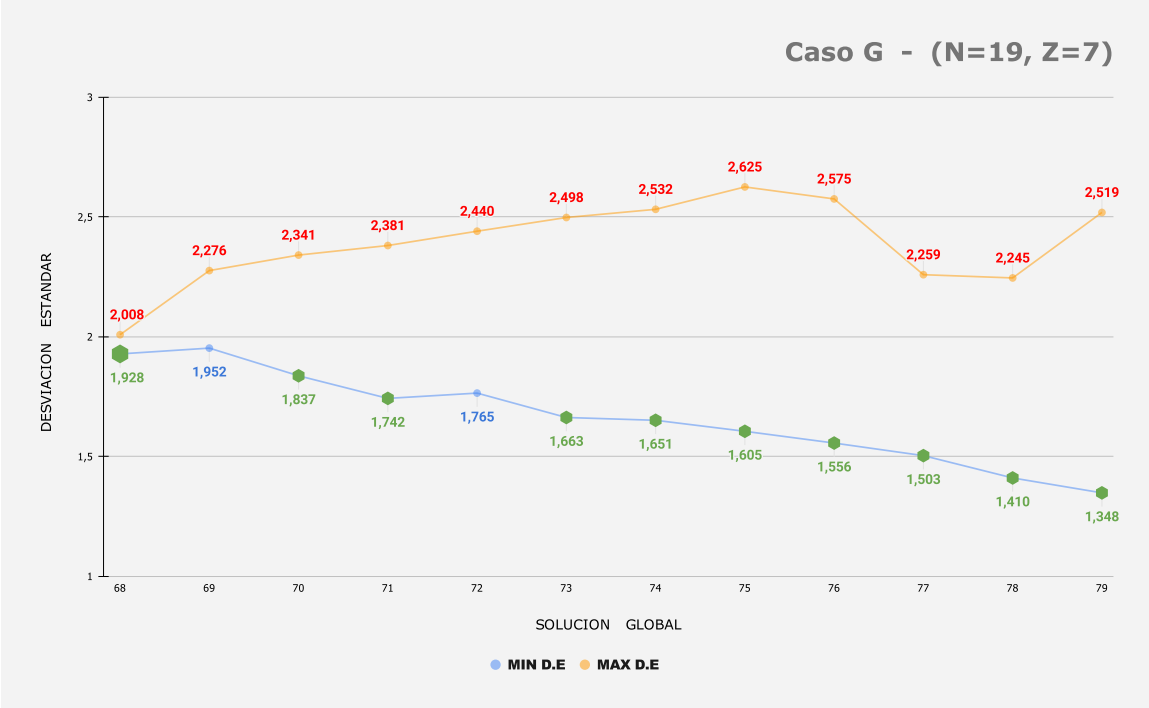


Figura 3.5: Frente de Pareto - [Caso G]

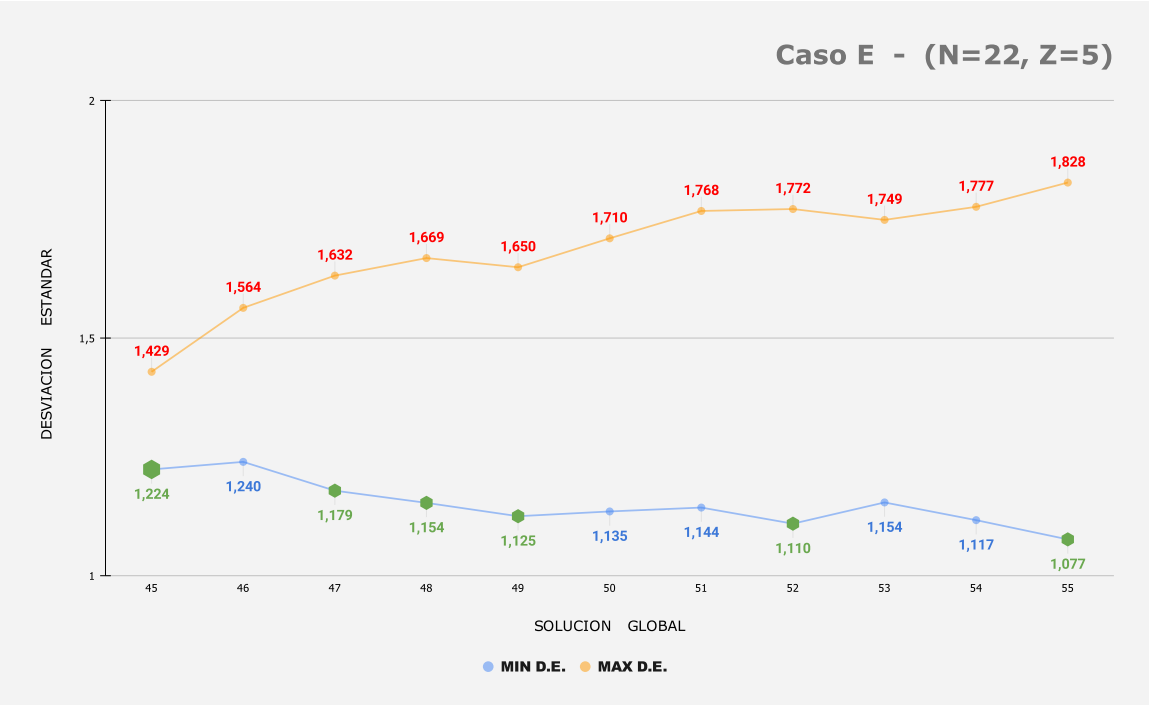


Figura 3.6: Frente de Pareto - [Caso E]

Estos gráficos son casos reales donde el valor de satisfacción global óptimo hallado contiene distintas posibilidades de asignación y contiene, en términos de equidad, una mejor y una peor. En algunos casos esta diferencia, aún mejor que nada, podría considerarse despreciable ya que la mejora es de aproximadamente un 4 % sobre el valor de equidad. Sin embargo, particularmente en el caso E, la diferencia de valor de equidad en el óptimo de satisfacción global no es menor (hay un 17 % de diferencia entre la peor y la mejor solución aún restringiendo al valor óptimo de la satisfacción global) y presenta una nueva posibilidad para MTAV. Esta asignación, que es un caso real de una cooperativa, fué realizada en Setiembre de 2021 y la asignación hallada y retornada por el sistema (por lo tanto la cual fué adoptada por los cooperativistas en aquella ocasión) fue aquella de peor equidad. Vea ahora cómo estas dos posibles asignaciones varían entre sí:

Caso E - 22 unidades																						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
MAX	5	1	1	1	2	4	1	1	1	1	2	2	1	3	1	2	1	5	5	3	1	1
MIN	2	1	1	2	2	4	2	1	1	1	2	2	2	3	1	2	1	5	5	3	1	1

Tabla 3.3: Resultados del análisis de equidad - I

Se destacan con el sombreado los ajustes encontrados durante la búsqueda de mejor equidad. En este caso, sin perder en cuanto a satisfacción promedio, se logra que tres cooperativistas cedan un punto cada uno en la escala de sus prioridades personales para que un cooperativista que estaba obteniendo la prioridad 5 (la peor posible según la cota superior Z aplicada para este sorteo), pueda mejorar su situación individual y obtener su segunda mejor opción de vivienda.

En las Tablas 3.4 y 3.5 se detallan las otras ocasiones de oportunidades similares:

Caso L - 13 unidades													
	1	2	3	4	5	6	7	8	9	10	11	12	13
MAX	2	4	3	3	4	5	3	1	3	4	5	5	1
MIN	2	1	3	3	5	4	4	4	3	4	5	4	1

Tabla 3.4: Resultados del análisis de equidad - II

Caso G - 19 unidades																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
MAX	1	1	3	4	6	3	1	7	5	6	5	1	1	5	5	5	3	5	1
MIN	1	4	2	4	6	3	1	7	5	6	5	1	3	1	5	5	3	5	1

Tabla 3.5: Resultados del análisis de equidad - III

Estos otros casos contienen algunos cambios opuestos entre sí, los cuales no implican una mejora para la equidad por sí mismos pero dan lugar a otros que sí lo hacen. En resumen para el caso L, se obtiene un cooperativista pasando de su prioridad 3 a su prioridad 4 y otro que cambia de 5 a 4 también. Luego, por último en el caso G, dentro de los cuatro valores que se modifican, se puede ver que la pareja 3 y 1 se mantiene, por lo tanto, los cambios netos son la dupla 1 y 5 de prioridades asignadas que se cambia por 2 y 4, lo cual resulta un poco más equitativo en términos generales.

3.7. Introducción del concepto de Vecindad

En esta sección se expone una breve introducción sobre cómo es posible realizar una extensión a la representación de la realidad de las cooperativas. Se propone una forma de contemplar la vecindad como parámetro complementario por parte de los usuarios y se formula un objetivo compuesto en conjunto con las preferencias de viviendas ya conocidas. Por último se aplica la implementación de dicha propuesta sobre un caso de ejemplo y se mencionan algunas observaciones.

3.7.1. Incluyendo la noción de Vecindad

Tras múltiples experiencias de cooperativistas que hicieron uso de MTAV, en algunas de las instancias, los usuarios mencionaron una propuesta que consiste en incluir una vía adicional de prioridades para lograr plasmar sus preferencias con mayor completitud. Por encima de un matching entre familias y hogares, lo que se busca resolver en la realidad de las cooperativas es una distribución dentro del complejo habitacional. Los usuarios hicieron referencia concretamente a la posibilidad de definir vecindades preferidas, es decir, poder dar conocimiento al sistema sobre estos deseos o necesidades que están ligadas a otros usuarios y no al inmueble.
Teniendo en consideración por un lado la situación anterior y por otro el nuevo paradigma utilizado como solución, se plantea llevar a cabo una prueba sobre la incorporación de estos parámetros complementarios al algoritmo.

La vecindad significa una manera adicional de alterar nuestro valor objetivo. Resulta que no es justamente trivial llegar a una definición práctica de este concepto de vecindad debido a que el mismo podría combinarse de distintas maneras en la función objetivo. Al mismo tiempo, es necesario establecer un grado de impacto o relevancia de este aspecto sobre la solución final relativo al que presentan las preferencias de viviendas.

El primer aspecto a definir es el modo en que los usuarios podrían representar sus preferencias de vecinos. Claramente la idea intuitiva es a través de distintas ponderaciones asociadas a cada uno de los demás cooperativistas, y la forma más natural de visualizar esta puntuación es asignando un valor entero de ganancia por el cumplimiento con cada una de las vecindades deseadas. La ponderación puede cuantificarse con un criterio como el escogido por MTAV para las preferencias de viviendas (donde 1 es el valor más buscado) o de forma opuesta, asignando mayor valor a quienes más se deseen tener por vecino. Para esta prueba, se considera el último caso, donde luego a mayor puntuación de vecindad, mayor ganancia representará en la función objetivo. Dado que la función objetivo actual es minimizada, se deberá substraer la bonificación obtenida por vecindad en caso de que corresponda.

En relación al nivel de influencia de este nuevo componente sobre el objetivo final, es deseable contemplar un balance coherente con respecto a los otros parámetros de los usuarios: ¿Debería permitirse que este valor tenga mayor repercusión que la preferencia sobre la vivienda?. La anterior es una pregunta importante si se tiene presente que esta ponderación de vecindad de carácter individual (es configurada por cada usuario independientemente), luego no solo afecta globalmente al valor que se busca optimizar (lo cual por sí solo puede significar una fuerte inclinación del algoritmo hacia un contexto puntual de vecinos) sino que también tiene implicancias sobre los otros cooperativistas. En un hipotético caso donde un usuario X prefiere viviendas del sector A, y otro usuario Y prefiere viviendas del sector opuesto B, si éste último tiene una fuerte preferencia de vecindad con el usuario X, luego el algoritmo debería tener cierto cuidado de no decantar fácilmente en ubicar al usuario X en el sector B solamente porque el usuario Y lo quiere por vecino y dispone un beneficio significativo para el objetivo general del programa por cumplir con ello. Si bien quizás este caso es un tanto extremo, en la suma se genera la motivación por limitar las opciones de ponderación.

Para mitigar a grandes rasgos la situación anterior, los valores de bonificación por vecindad no pueden dejarse libres y es fundamental fijar un rango válido de valores posibles.

Al incorporar esta variante, el problema se vuelve un problema no lineal, más precisamente un problema

de asignación cuadrático (QAP) [1] ya que la función objetivo tiene términos donde las variables de decisión aparecen multiplicadas entre sí. Esta nueva versión del modelo añade entonces dos parámetros extra, el primero será una matriz indicativa de cuales viviendas son vecinas o adyacentes y luego el segundo parámetro serían las entradas de los usuarios al respecto de la vecindad. Se lista a continuación la solución conceptual:

Parámetros:

n : Número de cooperativistas (y viviendas).
p_{ij} : Preferencia del cooperativista i sobre la vivienda j.
v_{ij} : Preferencia de vecindad del cooperativista i con el cooperativista j.
ad_{ij} : Booleano indicativo de si las viviendas i, j son consideradas adyacentes.

Variables de Decisión:

x_{ij} : Variable de decisión binaria que indica si el socio i es asignado a la vivienda j.

Problema de asignación de viviendas - Vecindad

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n} p_{ij} \cdot x_{ij} \; + \; \sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{r=1}^{n}\sum_{s=1}^{n} -(v_{kl} \cdot ad_{rs} \cdot x_{kr} \cdot x_{ls}) \tag{3.1}$$

$$\text{Subject to} \quad \sum_{j \in n} x_{k,j} = 1, \; \forall k \in n \tag{3.2}$$

$$\sum_{k \in n} x_{k,j} = 1, \; \forall j \in n \tag{3.3}$$

$$\sum_{j \in n} p_{k,j} x_{k,j} \; \leq \; Z, \; \forall k \in n \tag{3.4}$$

$$x_{k,j} \in 0, 1$$

Teniendo la nueva solución definida, los cambios correspondientes al modelo de Constraint Programming son relativamente directos. La nueva versión resulta entonces en la siguiente:

Modelo Extendido con Vecindad - Búsqueda de Satisfacción Global

---

```
include "alldifferent.mzn";
 ...
 ...

```

[1] Quadratic Assignment Problem - https://optimization.cbe.cornell.edu/index.php?title=Quadratic_assignment_problem

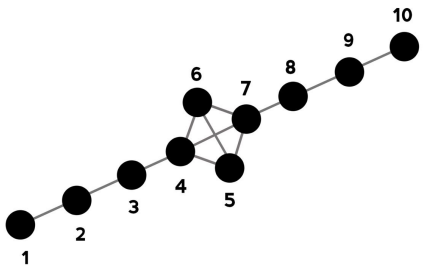
```
array[NUCLEOS,VIVIENDAS] of int: p; % prioridades %
array[NUCLEOS,PREFS] of int: ov; % viviendas ordenadas por prioridad %
array[VIVIENDAS,VIVIENDAS] of Bool: vady; % viviendas adyascentes %
array[NUCLEOS,NUCLEOS] of int: pv; % preferencias de vecindades %

array[NUCLEOS] of var PREFS: p_asignadas;  % de nucleo - prioridad
    asignada %
array[NUCLEOS] of var VIVIENDAS: v_asignadas; % de nucelo - vivienda
    asignada %
array[NUCLEOS] of var int: vecindad; % beneficio de vecindad para nucleo %

var int: satisf = sum(f in NUCLEOS)( p_asignadas[f] - vecindad[f] );


% Asignaciones all different %
constraint forall(f in NUCLEOS) (v_asignadas[f] = ov[f,p_asignadas[f]]);
constraint alldifferent(v_asignadas);

% Cota de prioridad maxima Z %
constraint z >= max(p_asignadas);

% Vecindad de un Nucleo %
constraint forall(f1 in NUCLEOS) (
  vecindad[f1] =
    sum(f2 in NUCLEOS)(
        vady[v_asignadas[f1], v_asignadas[f2]] * pv[f1, f2]
    )
 );

solve :: int_search(p_asignadas, varselect, valselect, complete)
    minimize satisf;
```

Listing 3.8 Minizinc - Vector de Prioridades (V2) - Vecindad

3.7.2. Análisis sobre un caso de prueba ejemplo

Como caso de ejemplo se toma el caso real I con 10 cooperativistas y viviendas. Para completar la realidad ejemplo, considere una disposición del complejo como la de la Figura 3.7 y su matriz de adyacencia asociada:

Se establece 0 a 3 como el rango de posibles puntuaciones de vecindad para esta asignación. Finalmente suponiendo que los cooperativistas tuvieran preferencias de vecindad tales que las familias 1 y 2 indican una preferencia de valor 2 entre sí porque son familia, misma situación que las familias 6 y 8 y por otra parte las familias amigas 2, 5 y 6 se puntúan con 1, se tiene la situación de la Figura 3.9 .



$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 3.7: Disposición y Matriz de adyacencia

$$\begin{bmatrix} 3 & 4 & 7 & 10 & 9 & 8 & 2 & 5 & 6 & 1 \\ 6 & 3 & 1 & 4 & 5 & 2 & 8 & 9 & 10 & 7 \\ 2 & 1 & 8 & 10 & 9 & 7 & 5 & 3 & 4 & 6 \\ 2 & 1 & 7 & 9 & 10 & 8 & 4 & 6 & 5 & 3 \\ 3 & 1 & 7 & 9 & 10 & 8 & 4 & 6 & 5 & 2 \\ 2 & 1 & 4 & 9 & 10 & 5 & 3 & 6 & 7 & 8 \\ 8 & 7 & 6 & 9 & 10 & 5 & 1 & 3 & 4 & 2 \\ 6 & 2 & 8 & 9 & 10 & 7 & 5 & 4 & 3 & 1 \\ 1 & 2 & 7 & 10 & 9 & 8 & 6 & 5 & 4 & 3 \\ 4 & 3 & 2 & 5 & 6 & 1 & 8 & 10 & 9 & 7 \end{bmatrix}$$

Figura 3.8: Preferencias de Vivienda

$$\begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Figura 3.9: Preferencias de Vecindad

Las entradas en la matriz de la Figura 3.7, llámense $a_{i,j}$, indican que las viviendas i y j son adyacentes, formando así una matriz simétrica. La Figura 3.8 contiene la matriz ya conocida de preferencias de los usuarios (filas) sobre las distintas viviendas (columnas). Por ultimo, la Figura 3.9 de entradas $v_{i,j}$ representa las preferencias de vecindad de los distintos usuarios (i, filas) sobre los demás usuarios (j, columnas) y no necesariamente tiene que ser simétrica.
Con estos nuevos datos, se comparan entonces dos asignaciones. Por un lado, haciendo uso del modelo clásico, sin vecindad. Por otro, aplicando el nuevo modelo (ambos casos con la configuración max_regret & split para las selecciones). Los resultados se presentan en las Tablas 3.6 y 3.7.

Caso I - 10 unidades											
Cooperativista	1	2	3	4	5	6	7	8	9	10	-
Vivienda	7	5	8	2	10	3	6	9	1	4	-
Preferencia	2	5	3	1	2	4	5	3	1	5	31
Satisfacción total											31

Tabla 3.6: Análisis Vecindad - Resultados SIN Vecindad

Caso I - 10 unidades											
Cooperativista	1	2	3	4	5	6	7	8	9	10	-
Preferencia	2	5	3	2	1	4	5	1	4	5	32
Vivienda	7	5	8	1	2	3	6	10	9	4	-
Vecindad	✓	✓	-	-	✓	✓	-	-	-	-	-
Ganancia Vecinos	2	2	-	-	1	1	-	-	-	-	-6
Satisfacción total											26

Tabla 3.7: Análisis Vecindad - Resultados CON Vecindad

Observando las Tablas 3.6 y 3.7 se destaca que con los nuevos datos de vecindad la asignación es diferente, concluyendo en una mejor satisfacción a nivel grupal gracias a los beneficios obtenidos de fijar vecindades deseadas por los usuarios.
Finalmente resaltar que no todas las posibles vecindades ponderadas fueron correspondidas, habiendo quedado fuera la posibilidad de ubicar como vecinos a los socios 6 y 8.
Aunque el anterior es un ejemplo relativamente simple (por la cantidad reducida de viviendas y por las vecindades que significan un beneficio considerable respecto al valor total de satisfacción), de igual forma nos permite visualizar la nueva oportunidad que se genera y que trae la opción de continuar con su investigación en otros casos así como la posibilidad de extender la solución actual en un futuro.

Capítulo 4

Conclusiones

4.1. Avance logrado

Como resultado de la fase de investigación, se ha realizado un progreso significativo al explorar el enfoque de Constraint Programming (CP) como una alternativa valiosa para la resolución de problemas de asignación. Este enfoque ha demostrado ser altamente flexible en el modelado de preferencias de usuarios, permitiendo la representación de relaciones y funciones objetivo de manera más expresiva y precisa.

Uno de los aspectos más destacados del avance ha sido la consideración de la equidad en las asignaciones. Se propuso medir la equidad a través de la desviación estándar entre los valores de prioridad en las asignaciones resultantes y el análisis de esta métrica objetiva reveló que aún habían márgenes de mejora en las soluciones encontradas, siendo en algunos casos más significativo que en otros. Se observó también que en algunos casos puede llegar a ser útil considerar las distintas opciones de resultados pertenecientes al Frente de Pareto.

Además, se llevaron a cabo benchmarks exhaustivos comparando el enfoque de Constraint Programming con la Programación Matemática. Estos benchmarks han permitido evaluar la eficiencia y eficacia del nuevo enfoque en una variedad de escenarios y tamaños de problema.

Fueron utilizadas diferentes combinaciones de estrategias para la búsqueda en el espacio de soluciones por parte del solver. Tras una primera iteración en la que los resultados presentaban una brecha mayor con los de la versión de programación entera, se realizó otra selección de estrategias y se volvió a analizar cada uno de los casos. Entre estos últimos valores obtenidos, de forma general a lo largo de todos los casos testeados, se destacaron los asociados a una estrategia puntual: max_regret para la selección de variable y split o default-undefined para la selección de valor. Este set de estrategias reduce considerablemente, en la mayoría de los casos, los tiempos de las demás estrategias utilizadas. Aún en aquellos casos en los que otra estrategia presenta mejor desempeño, la eficiencia de max_regret puede considerarse suficiente.

Se llegó a la conclusión de que el desempeño logrado es relativamente bueno en la mayoría de los casos reales ejecutados, y en los casos más complejos, de todas maneras existe la posibilidad de apoyarse en la Programación Matemática para luego continuar la búsqueda en la tercera etapa donde se analiza la equidad de la solución óptima en cuanto a satisfacción global.

Por último brevemente se mostró el potencial que puede tener la extensión de la solución actual incluyendo preferencias alternas por parte de los usuarios, de modo que el algoritmo pueda considerar con mayor detalle los beneficios reales de optar por las distintas asignaciones y por lo tanto pueda concluir en mejores resultados finales. Utilizando un ejemplo concreto se implementó un modelo que contemple lo anterior, precisamente tomando el concepto de vecindad como extensión y añadiendo preferencias para ello. Se pudo apreciar la practicidad por parte de Constraint Programming para incorporar modificaciones en problemas del entorno discreto.

En conjunto, este trabajo ha llevado a reconocer la potencial robustez del enfoque de Constraint Programming para abordar este problema, resultado que fue brevemente compartido en la conferencia CLAIO

del año 2022 (Cancela, Fierro, Manera, y Prino, 2022) y en la cual, participando de la misma, también se conocieron otros casos donde se hizo uso del enfoque y se obtuvieron conclusiones interesantes.
El avance obtenido sienta las bases para investigaciones futuras y la implementación práctica de soluciones basadas en Constraint Programming en problemas de asignación prioritaria.

4.2. Futuras investigaciones y Propuestas de trabajo

A raíz de lo investigado y analizado en esta tesis surgen aspectos del entorno MTAV que son interesantes para continuar estudiando así como también nuevas propuestas de trabajo.

Incorporación del valor Equidad

Conociendo las posibilidades de mejora que hay en el análisis de equidad de las soluciones, resulta de interés continuar el trabajo realizado con el fin de incorporar este nuevo aspecto de la solución general a instancias futuras de sorteos reales de cooperativas, evaluando la recepción que esta nueva medida tenga por parte de los usuarios. Se podría tanto añadir la tercera etapa al mecanismo, buscando mejorar desde este punto de vista la solución hallada como también por otro lado, presentar la posibilidad (a través del Frente de Pareto) de escoger mayor equidad en pos de puntos de satisfacción global o viceversa.

Solvers y Performance

Es deseable también profundizar en las pruebas con diferentes solvers y características de los mismos así como también investigar si el modelo puede volverse más eficiente haciendo uso de estructuras de datos y variables alternativas. Un punto interesante sería realizar pruebas con solvers que contemplen Warm Startups - funcionalidad que permite inicializar las variables de decisión con una instancia específica de valores definida en los datos de entrada del modelo. Esto puede significar una mejora en los tiempos ya que previamente se recorre el espacio de soluciones al llevar a cabo la búsqueda de la cota Z en primera instancia y luego de la satisfacción global óptima. Teniendo las etapas de la solución general en modelos independientes impide que a priori se consiga beneficio alguno del trabajo realizado por etapas anteriores. Por otro lado existe también la posibilidad, con el solver CPLEX, de definir en un solo modelo las tres funciones objetivo utilizadas en las tres etapas y solicitar la optimización lexicográfica de los mismos [1]. Esta funcionalidad puede llegar a presentar resultados interesantes ya que el solver internamente se encarga de mantener el contexto de la ejecución mientras avanza de un objetivo a otro.

Expansión de la solución

La implementación alternativa elaborada en este trabajo es fácilmente escalable en cuanto a restricciones y estructuras de datos, con lo cual queda disponible una oportunidad de estudiar la combinación de distintos aspectos de la problemática asociada a la asignación de viviendas en las cooperativas. Por el pedido real de parte de usuarios, la principal idea sobre esto es analizar la vecindad entre familias como foco de preferencias paralelas que permitan a la herramienta, quizás, tener un mayor conjunto de soluciones. Por otra parte también se podría permitir la posibilidad de definir prioridades en bloque, que significa la capacidad de poder indicar con igual prioridad varias viviendas que son indiferentes para un cooperativista. Esta mejora flexibiliza tanto el ingreso de datos por parte de los usuarios al mismo tiempo que evitaría generar combinaciones

[1] CP Optimizer - minimizeStaticLex - https://www.ibm.com/docs/en/icos/22.1.1?topic=functions-minimizestaticlex

extra innecesarias en el espacio de soluciones del problema. Ambos casos requieren una reestructuración de los modelos ya que precisan datos y variables adicionales. Además, para el caso de preferencias en bloques, no se corresponderían uno a uno prioridades con viviendas como en el modelo presentado y entonces el conjunto de restricciones existentes sería también reescrito.

MTAV Online

A la fecha de hoy, MTAV se distribuye como aplicación de escritorio cuyo instalable se puede descargar públicamente junto con el manual de usuario. Si bien este factor no ha presentado inconvenientes significativos, podría resultar más práctico evitar este paso de instalación - preparación, y en cambio tener a disposición la herramienta online donde los cooperativistas puedan ingresar sus preferencias de forma individual. Este cambio presentaría múltiples ventajas en el uso y difusión de MTAV:

- Accediendo mediante un navegador de internet, se evitan dependencias técnicas entre la aplicación y el dispositivo con el que los usuarios disponen.
- Se podría hacer uso de la herramienta desde una gama de dispositivos ampliamente mayor ya que los dispositivos móviles (tipo de dispositivo más utilizado hoy día) también podría acceder al sitio donde MTAV estaría a disposición.
- El sitio podría también facilitar la generación de plantillas para el ingreso de preferencias, componente previo que suele implementarse manualmente por los usuarios en sistemas de planillas de cálculo como Excel o Google Sheets.
- La difusión de MTAV como tecnología desarrollada por la Facultad de Ingeniería para el uso libre sería más sencilla, simplemente compartiendo links del sitio web. A su vez, los manuales de usuario, documentaciones y testamentos de cooperativas que han usado previamente la solución tendrían mayor visibilidad. Por último, se accedería rápidamente a los diferentes medios de contacto con el equipo de MTAV y los cooperativistas voluntarios que ayudan a las nuevas cooperativas a dar los primeros pasos en la comprensión y uso de MTAV para sus asignaciones.
- Desde un punto de vista técnico, se obtendrían ventajas como disponer de acceso libre a todos los logs de las ejecuciones además de poder realizar mantenimientos con mayor dinamismo al software.

Un aspecto no menor también asociado a la diferencia entre utilizar el sistema desde un ordenador de escritorio o desde un servidor online es el factor de duda que los cooperativistas pueden tener a partir de lo desconocido. MTAV es una solución tecnológica con características avanzadas en la resolución del problema y esto puede generar desconfianza si la operación de dicha tecnología la realiza uno de los socios cooperativistas también participante de la asignación (o un grupo de ellos). Mencionado en la sección de Estudios Actuales, en su tesis de maestría, Joaquín Paleo demuestra que en casos puntuales el mecanismo no cumple con la propiedad de ser robusto frente a estrategias. Paleo da un ejemplo y concluye que si las condiciones se presentan, un usuario que tuviera la totalidad de la información sobre las preferencias ajenas, podría ajustar sus valores de entrada para beneficiarse en el resultado final. Este suceso podría considerarse prácticamente improbable. De todas maneras, si las preferencias se ingresaran en privado mediante el uso de usuarios y contraseñas para el acceso al servidor, la exposición de información personal no tendría lugar en primera instancia generando anonimato y tranquilidad en el manejo de información personal de cada cooperativista, y con ello, una mejor recepción de la herramienta por parte de todos.

Referencias

Cancela, H., Fierro, M., Manera, A., y Prino, M. (2022). Optimization methods for units' assignment in housing cooperatives. En *CLAIO 2022 - XXI Latin Ibero-American Conference on Operations Research.* Buenos Aires, Argentina.

de Harder, H. (2023). *Constraint programming explained.* Descargado de https://towardsdatascience.com/constraint-programming-explained-2882dc3ad9df (último acceso en 30.04.2024)

Dimény, I., y Koltai, T. (2023). Comparison of MILP and CP models for balancing partially automated assembly lines. *Central European Journal of Operations Research*, *10.1007*. doi: https://doi.org/10.1007/s10100-023-00885-x

Engquist, M. (1980). *A successive shortest path algorithm for the assignment problem.* Defense Technical Information Center.

Fagián, I., Prino, M., y Sánchez, E. (2017). *Informe módulo de taller: Módulo viviendas - aplicación con interfaz gráfica.* (Supervisado por Cancela, Héctor. Facultad de Ingeniería, Universidad de la República)

Fierro, M. (2020). *Informe módulo de extensión: Mtav 3.0.* (Supervisado por Cancela, Héctor. Facultad de Ingeniería, Universidad de la República)

FUCVAM. (s.f.). *Historia.* Descargado de https://www.fucvam.org.uy/quienes-somos/historia (último acceso en 31.03.2024)

Housing Europe. (2012). *Cooperative housing: A key model for sustainable housing in Europe.* Descargado de https://www.housingeurope.eu/event-183/cooperative-housing (último acceso en 31.03.2024)

IBM. (2022). *Overview of constraint programming.* Descargado de https://ibmdecisionoptimization.github.io/docplex-doc/cp.html (último acceso en 30.04.2024)

Korte, B. H., Vygen, J., Korte, B., y Vygen, J. (2011). *Combinatorial optimization* (Vol. 1). Springer.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*(1-2), 83-97. Descargado de https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109 doi: https://doi.org/10.1002/nav.3800020109

Long, X. (2016). *Essays on house allocation problems* (Tesis Doctoral, Texas A & M University). (último acceso en 31.03.2024) doi: https://hdl.handle.net/1969.1/157810

Mears, C., Schutt, A., Stuckey, P. J., Tack, G., Marriott, K., y Wallace, M. (2014). Modelling with option types in minizinc. En H. Simonis (Ed.), *Integration of ai and or techniques in constraint programming* (pp. 88–103). Cham: Springer International Publishing.

Mehdi Khosrow-Pour, D. (2017). *Encyclopedia of information science and technology, 4th edition* (Vol. Swarm Intelligence for Multi-Objective Optimization in Engineering Design). Information Science Reference.

Michael R. Garey, D. S. J. (1979). *Computers and intractability: A guide to the theory of np-completeness* (Vol. 1). W. H. Freeman.

Najman, J. (2023). *What are the differences between Constraint Programming and MIP?* (último acceso en 31.03.2024) doi: https://support.gurobi.com/hc/en-us/articles/360048197891-What-are-the-differences-between-Constraint-Programming-and-MIP

Paleo Arrarte, J. M. (2021). *La asignación de apartamentos en cooperativas de vivienda : un enfoque desde el diseño de mercados.* (Tesis de Maestría - Udelar, FCEA, último acceso en 31.03.2024) doi: https://www.colibri.udelar.edu.uy/jspui/handle/20.500.12008/31185

Prino, M., Sánchez, E., y Cancela, H. (2016). Optimal distribution of habitational units in a cooperative: A mathematical application to optimize satisfaction. En *2016 XLII Latin American Computing Conference (CLEI)* (p. 1-7). doi: 10.1109/CLEI.2016.7833357

Rossi, F., van Beek, P., y Walsh, T. (Eds.). (2006). *Handbook of constraint programming* (Vol. 2). Elsevier.

Sánchez-Laulhé Sánchez de Cos, J. M., Rabasco Pouzelo, P., y Solanas Domínguez, M. (2013). Cooperativas de vivienda: transferencias imposibles suecia- uruguay- españa. En *Procesos extremos en la constitución de la ciudad. de la crisis a la emergencia en los espacios mundializados* (p. 110-121). Descargado de https://idus.us.es/bitstream/handle/11441/52403/COPERATIVA%20DE%20VIVIENDAS.pdf?sequence=1&isAllowed=y

Wheeler, A., Wei, C., Bahadir, C. D., Shui, R., y Zhang, Z. (2020). *Network flow problem.* Descargado de https://optimization.cbe.cornell.edu/index.php?title=Network_flow_problem (Material del curso: ChemE 6800, Cornell University. Último acceso en 31.03.2024)

Zou, X., Fang, S., Huang, Y., y Zhang, L. (2017). Mixed-integer linear programming approach for scheduling repetitive projects with time-cost trade-off consideration. *J.Comput. Civ. Eng.*, *31*(3). doi: https://doi.org/10.1061/(asce)cp.1943-5487.0000641

Zou, X., y Zhang, L. (2020). A constraint programming approach for scheduling repetitive projects with atypical activities considering soft logic. *Automation in Construction*, *109*, 102990. doi: https://doi.org/10.1016/j.autcon.2019.102990

Çakır, G., Subulan, K., Yildiz, S. T., Hamzadayı, A., y Asılkefeli, C. (2022). A comparative study of modeling and solution approaches for the multi-mode resource-constrained discrete time–cost trade-off problem: Case study of an erp implementation project. *Computers & Industrial Engineering*, *169*, 108201. doi: https://doi.org/10.1016/j.cie.2022.108201

Özbakir, L., Baykasoğlu, A., y Tapkan, P. (2010). Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation*, *215*(11), 3782-3795. Descargado de https://www.sciencedirect.com/science/article/pii/S0096300309010078 doi: https://doi.org/10.1016/j.amc.2009.11.018

Anexo A

Anexos

A.1. Aspectos técnicos

En este capítulo se describe el ambiente utilizado junto a los software y sus versiones utilizadas a lo largo de la etapa de investigación y desarrollo del trabajo presente.

Durante todo el proceso, las ejecuciones de algoritmos fueron realizadas en un computador personal *MacBook Pro* de las siguientes características:

- Procesador: - 2,2 GHz 6-Core Intel Core i7
- Memoria: - 16 GB 2400 MHz DDR4
- Gráficos: - Radeon Pro 555X 4 GB
- Disco: - 256GB Flash Storage (SSD)

Para las comparaciones con la solución actual disponible de MTAV se utilizaron los modelos matemáticos incluidos en la aplicación de software en lugar de la aplicación en sí misma. Los mismos fueron puestos a disposición por parte del equipo a cargo de MTAV y la ejecución de las búsquedas fueron realizadas con **GLPSOL: GLPK LP/MIP Solver, v4.65**. El comando utilizado en todas las ocasiones fue el siguiente:

glpsol --model <my_model.mod> --data <my_data.dat>
--output <output_file.sol> --tmlim 75 --log <log_file.log>

En lo que a Constraint Programming respecta, los modelos se escribieron utilizando el lenguaje MiniZinc y ejecutados con la aplicación de línea de comando también MiniZinc (MiniZinc to FlatZinc converter, version 2.7.1 - Copyright (C) 2014-2023 Monash University, NICTA, Data61). La versión del solver seleccionado - Gecode - para la posterior ejecución de los casos es la 6.2.0.

Por último, se utilizó Elixir en su versión 1.14.3 (compilado con Erlang/OTP 25) como lenguaje de programación en los scripts implementados para:

- Generar las matrices de viviendas ordenadas por preferencia
- Construir los Frentes de Pareto iterando sobre la satisfacción global.

Todos los archivos relacionados a este trabajo se encuentran públicos en el repositorio https://gitlab.fing.edu.uy/marcos.fierro/tesis.

A.2. Casos de prueba sobre MTAV3.0

A continuación se detallan las prioridades asociadas a cada caso de prueba utilizado:

Caso A - 5 unidades

F \ V	1	2	3	4	5
1	5	2	3	1	4
2	4	3	2	1	5
3	5	4	3	2	1
4	4	3	2	1	5
5	5	1	2	3	4

Caso B - 20 unidades

F \ V	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	20	2	13	3	19	18	17	7	16	15	14	12	11	6	10	4	5	9	1	8
2	20	4	5	3	19	14	15	16	17	11	12	13	18	9	10	8	6	7	2	1
3	19	2	3	1	17	7	8	4	18	9	10	5	20	11	12	6	13	14	15	16
4	20	19	18	17	13	14	15	16	12	11	10	3	9	6	7	8	1	2	4	5
5	20	5	6	8	19	12	13	14	15	16	10	9	18	17	7	11	4	3	2	1
6	20	19	18	17	16	15	14	2	13	1	7	11	12	3	5	8	4	6	9	10
7	20	2	11	1	18	4	15	3	17	7	16	5	19	8	12	6	9	13	10	14
8	13	14	15	7	10	16	20	17	9	18	19	6	8	11	12	5	4	2	3	1
9	19	7	8	2	18	15	16	4	17	13	14	1	20	11	12	3	9	10	5	6
10	15	13	14	1	11	3	12	2	18	7	8	4	17	9	10	16	5	6	19	20
11	17	8	9	4	18	6	7	5	19	3	1	2	20	11	12	10	14	13	15	16
12	15	17	18	16	14	7	12	6	13	3	4	5	20	10	9	19	1	8	2	11
13	20	3	2	1	17	12	16	6	18	11	15	5	19	10	14	4	9	13	7	8
14	19	4	9	1	18	5	11	2	17	6	12	3	20	7	13	10	8	16	15	14
15	20	2	1	18	19	16	15	10	11	12	13	9	17	8	14	5	7	6	3	4
16	20	1	5	4	19	16	18	8	3	15	17	7	2	13	14	6	11	12	9	10
17	11	2	3	1	12	8	16	13	9	7	17	14	10	6	18	15	5	19	4	20
18	20	3	1	2	19	16	15	14	18	12	13	11	17	8	9	10	7	6	4	5
19	20	1	3	2	19	4	8	12	18	11	5	13	17	14	15	16	6	9	7	10
20	11	3	2	1	9	15	20	6	10	14	19	5	12	13	18	4	8	17	7	16

Caso C - 22 unidades

F \ V	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	22	20	19	1	3	2	21	18	12	6	15	5	17	11	8	14	7	16	4	10	13	9
2	22	21	20	1	3	2	19	18	17	6	5	4	16	15	7	8	9	14	13	12	10	11
3	22	21	20	1	19	18	17	3	2	16	15	14	5	4	6	7	8	9	10	11	12	13
4	20	21	9	1	3	2	16	22	10	4	13	5	17	11	6	14	7	18	12	19	15	8
5	3	17	2	1	18	4	5	19	6	7	20	8	9	11	15	21	13	10	12	16	22	14
6	12	10	14	2	1	5	11	3	13	6	19	22	4	15	7	18	21	9	16	8	17	20
7	11	20	10	4	5	2	9	18	6	14	15	22	17	1	8	21	13	16	3	7	19	12
8	22	21	20	1	4	2	19	18	14	15	17	16	13	9	8	10	5	12	3	6	11	7
9	9	8	7	22	21	20	10	6	5	19	18	17	4	3	14	15	16	2	1	13	12	11
10	17	7	4	3	2	1	10	8	5	13	15	11	9	6	14	16	12	19	18	20	22	21
11	21	20	19	3	1	2	18	17	16	15	10	11	22	12	9	7	8	14	13	6	4	5
12	22	21	20	2	3	1	19	18	17	4	6	5	16	15	7	9	8	14	13	10	12	11
13	22	13	20	10	7	8	21	16	11	12	19	9	15	1	6	18	5	14	2	3	17	4
14	21	12	9	1	3	2	20	17	18	22	8	19	16	6	11	10	7	15	4	14	13	5
15	10	19	16	2	3	6	7	15	4	13	14	20	8	1	9	12	21	17	5	18	22	11
16	22	21	20	1	2	3	19	17	18	9	13	14	15	16	12	11	10	8	7	6	5	4
17	22	18	8	3	2	1	9	17	7	14	21	10	15	6	12	20	11	16	5	4	19	13
18	22	21	20	2	3	1	19	14	11	13	4	12	18	9	8	17	5	16	10	7	15	6
19	21	14	15	1	3	2	11	12	13	22	4	5	9	10	6	7	8	16	17	19	20	18
20	17	13	12	2	4	3	16	9	8	18	19	20	7	6	14	21	15	5	1	22	11	10
21	22	8	4	3	1	2	12	9	5	15	21	18	10	6	14	20	17	11	7	13	19	16
22	22	20	19	7	6	5	21	12	11	18	17	16	4	3	15	14	13	2	1	10	9	8

Caso N - 29 unidades

F \ V	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	
1	1	4	18	2	5	19	6	8	20	9	15	23	10	14	26
	27	28	29	11	17	24	12	16	25	3	13	22	7	21	
2	17	18	28	1	5	21	2	6	22	9	13	24	10	14	25
	7	8	23	11	15	26	12	16	27	19	4	20	3	29	
3	12	13	14	1	2	9	3	4	10	21	22	23	24	25	26
	27	28	29	15	16	19	17	18	20	5	6	11	7	8	
4	28	17	10	24	19	4	23	18	3	22	13	6	26	15	9
	27	16	7	25	14	8	21	12	5	29	11	1	20	2	
5	3	1	2	6	4	5	12	10	11	19	17	18	22	20	21
	29	15	16	28	26	27	25	23	24	9	7	8	13	14	
6	7	8	14	3	4	15	2	16	17	9	18	19	10	20	21
	11	22	23	12	24	25	13	26	27	1	5	28	6	29	

7	16	17	26	1	7	20	2	3	18	9	10	24	12	15	25
	29	28	27	11	14	23	8	13	22	4	5	19	6	21	
8	27	16	15	25	12	11	22	4	3	29	20	19	28	18	17
	26	14	13	23	6	5	24	8	7	21	2	1	10	9	
9	5	10	16	4	6	13	2	7	14	18	23	26	19	22	27
	17	1	11	21	24	29	20	25	28	3	8	12	9	15	
10	29	27	28	17	18	19	8	1	5	24	25	26	20	6	7
	21	22	23	13	4	14	11	3	12	9	2	10	15	16	
11	28	26	27	25	6	18	3	1	16	7	9	20	8	10	21
	29	15	24	14	12	23	13	11	22	4	2	17	5	19	
12	3	1	2	14	10	11	13	8	9	19	15	17	20	16	18
	29	21	22	27	24	26	28	23	25	12	6	7	4	5	
13	26	10	9	25	18	17	24	1	2	23	15	16	22	20	19
	21	14	13	27	4	3	28	6	5	29	12	11	7	8	
14	4	8	17	3	7	16	1	5	14	21	23	25	22	24	26
	27	28	29	10	12	19	11	13	20	2	6	15	9	18	
15	18	16	17	15	14	7	13	12	6	21	19	20	26	24	25
	29	28	27	23	10	11	22	9	8	5	4	3	2	1	
16	20	2	4	26	11	16	21	5	6	24	9	14	25	10	15
	29	27	28	23	8	13	22	7	12	19	1	3	17	18	
17	16	15	17	10	9	11	7	6	8	25	24	26	28	27	29
	13	12	14	19	18	20	21	22	23	2	1	3	4	5	
18	21	6	1	26	13	11	28	17	15	24	9	4	25	10	5
	29	20	19	23	8	3	22	7	2	27	14	12	18	16	
19	8	9	13	5	6	12	1	4	10	14	18	21	15	19	22
	25	27	28	16	20	23	17	24	26	29	2	11	3	7	
20	11	20	1	19	28	9	16	25	6	13	22	3	12	21	2
	17	26	7	14	23	4	15	24	5	18	27	8	29	10	
21	25	26	27	22	23	24	7	8	9	3	4	6	1	2	5
	13	14	15	16	17	18	20	19	21	10	11	12	28	29	
22	14	6	11	5	3	4	13	8	10	23	15	22	24	16	21
	26	27	28	29	17	20	25	18	19	12	7	9	1	2	
23	5	6	13	20	21	25	16	18	23	1	2	11	3	4	12
	29	27	28	8	10	15	7	9	14	17	19	24	22	26	

24	26	25	24	11	8	4	10	6	2	23	19	20	22	18	21
	29	28	27	16	13	14	17	12	15	9	5	1	7	3	
25	8	9	24	7	6	17	2	1	14	11	10	25	13	12	26
	19	18	27	21	20	28	23	22	29	4	16	3	15	5	
26	28	24	26	29	25	27	22	16	17	12	6	8	11	5	7
	15	13	14	9	1	3	10	2	4	23	18	19	20	21	
27	12	17	22	1	5	9	2	6	10	13	18	23	14	19	24
	27	28	29	15	20	25	16	21	26	3	7	11	4	8	
28	29	27	28	6	4	5	3	2	1	26	24	25	23	21	22
	8	7	9	16	15	17	20	18	19	12	10	11	13	14	
29	17	16	26	15	14	25	4	3	19	7	6	21	13	12	24
	28	27	29	11	10	23	9	8	22	2	1	18	5	20	

Caso H - 4 unidades

F \ V	**1**	**2**	**3**	**4**
1	2	4	3	1
2	2	3	4	1
3	4	3	2	1
4	2	3	4	1

Caso I - 10 unidades

F \ V	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	3	4	7	10	9	8	2	5	6	1
2	6	3	1	4	5	2	8	9	10	7
3	2	1	8	10	9	7	5	3	4	6
4	2	1	7	9	10	8	4	6	5	3
5	3	1	7	9	10	8	4	6	5	2
6	2	1	4	9	10	5	3	6	7	8
7	8	7	6	9	10	5	1	3	4	2
8	6	2	8	9	10	7	5	4	3	1
9	1	2	7	10	9	8	6	5	4	3
10	4	3	2	5	6	1	8	10	9	7

Caso J - 14 unidades

F \ V	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
1	7	10	12	13	2	4	6	5	3	1	8	9	11	14
2	10	13	14	8	6	1	2	3	4	5	9	12	11	7
3	1	4	8	11	5	14	12	6	3	2	7	9	10	13
4	11	13	14	12	4	3	1	2	5	6	8	9	10	7
5	2	6	7	5	9	10	3	8	4	1	11	13	14	12
6	5	13	11	14	12	9	2	1	8	7	3	10	6	4
7	10	12	13	14	6	3	1	8	4	2	9	11	7	5
8	11	12	13	14	3	6	2	1	4	7	10	9	8	5
9	14	12	13	11	3	5	9	8	6	2	4	1	7	10
10	14	13	12	11	10	6	3	7	2	4	5	8	9	1
11	13	12	11	10	9	3	2	7	8	14	4	5	6	1
12	9	13	14	10	4	5	3	6	2	1	7	11	12	8
13	14	13	12	11	4	10	3	2	1	5	9	8	7	6
14	10	11	12	13	7	6	5	4	3	2	14	1	9	8

Caso D - 11 unidades

F \ V	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	10	9	11	2	1	3	6	5	4	8	7
2	9	10	11	4	3	7	8	2	1	5	6
3	11	10	3	5	4	2	1	9	8	7	6
4	10	9	11	1	2	5	6	4	3	7	8
5	4	1	11	3	2	6	5	8	7	10	9
6	7	6	5	3	4	2	1	8	9	10	11
7	7	1	6	5	4	3	2	8	9	11	10
8	11	10	3	5	4	1	2	8	7	6	9
9	10	9	3	5	6	2	1	8	7	4	11
10	4	8	9	1	2	6	5	3	7	11	10
11	6	4	1	7	5	2	3	11	10	8	9

Caso E - 22 unidades

F \ V	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	14	15	20	6	4	1	16	17	21	10	5	8	7	3	2	18	19	22	11	12	9	13
2	4	5	1	6	7	8	9	10	2	18	17	16	15	14	13	12	11	3	22	21	20	19
3	1	2	20	3	14	15	18	19	21	10	5	4	8	16	17	12	13	22	11	7	6	9
4	20	21	22	12	9	15	6	7	13	17	2	1	11	10	16	5	8	14	18	4	3	19
5	12	13	22	7	4	3	6	11	21	20	18	9	8	2	1	5	16	10	19	14	15	17
6	15	14	16	7	22	21	8	9	20	10	11	12	3	19	18	4	13	17	6	5	2	1
7	17	16	22	9	11	13	6	5	21	20	15	2	8	10	12	4	3	19	18	14	1	7
8	20	21	22	8	4	1	17	18	9	6	12	11	5	3	2	13	14	10	7	19	15	16
9	21	22	20	13	6	3	12	18	1	8	14	10	7	5	4	11	19	2	9	15	16	17
10	4	3	20	12	18	19	6	5	9	1	2	13	11	22	21	8	7	10	16	17	14	15
11	22	6	12	3	7	1	2	5	4	15	16	14	13	9	8	11	17	10	18	19	20	21
12	13	12	22	7	2	3	5	6	21	11	10	9	8	1	4	14	15	20	19	18	17	16
13	19	21	22	11	4	6	16	17	18	13	10	5	14	3	1	7	20	15	12	9	2	8
14	4	1	22	7	6	5	2	3	8	17	16	10	9	14	13	15	21	18	20	19	12	11
15	2	1	22	7	9	8	6	4	20	3	5	11	10	15	16	14	13	21	17	12	18	19
16	11	18	14	4	3	8	12	21	15	1	19	6	5	7	9	13	22	16	2	20	17	10
17	22	21	20	7	9	8	3	4	18	15	12	2	1	11	10	6	5	19	16	14	13	17
18	18	17	16	5	2	4	11	13	14	9	20	8	6	1	3	10	12	7	15	19	21	22
19	10	13	6	16	1	4	9	15	7	22	20	18	12	2	3	8	14	5	21	19	17	11
20	10	9	20	8	18	19	17	16	21	7	11	5	13	14	15	4	12	22	6	3	1	2
21	14	15	22	9	18	19	12	13	21	8	6	5	3	16	17	10	11	20	7	4	2	1
22	13	5	22	14	12	11	7	3	21	16	6	8	18	10	9	4	1	20	15	2	17	19

Caso F - 18 unidades

F \ V	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	1	2	18	3	4	5	6	17	7	8	9	10	16	11	12	13	14	15
2	14	13	18	12	3	9	6	17	11	2	8	5	16	10	1	7	4	15
3	17	16	18	12	9	11	10	15	7	1	3	5	14	8	2	4	6	13
4	16	17	18	14	11	13	12	15	3	1	4	2	5	7	6	9	8	10
5	13	14	18	11	3	6	9	17	10	1	4	7	15	12	2	5	8	16
6	4	1	2	18	17	11	3	6	16	15	10	5	7	14	13	9	8	12
7	3	8	18	7	5	10	6	17	1	11	13	2	16	4	14	9	12	15
8	13	14	18	12	10	9	11	17	5	3	6	8	16	1	2	4	7	15
9	8	11	18	2	12	5	14	17	3	1	6	9	16	4	13	7	10	15
10	14	13	18	6	11	7	5	17	9	12	10	8	16	2	4	3	1	15
11	8	7	17	14	1	2	3	18	9	4	5	6	15	11	10	12	13	16
12	3	6	18	7	8	1	2	17	9	10	4	5	16	13	14	11	12	15
13	14	13	18	12	3	9	6	17	10	1	7	4	16	11	2	8	5	15
14	16	17	18	9	3	10	6	14	12	1	8	4	15	13	2	7	5	11
15	11	12	16	13	8	9	10	15	4	3	6	17	18	2	1	5	7	14
16	2	1	18	4	3	5	6	15	8	7	9	10	16	12	11	13	14	17
17	13	14	18	5	3	9	12	17	6	2	8	11	16	4	1	7	10	15
18	13	14	18	3	9	6	12	17	2	8	5	11	16	1	7	4	10	15

Caso G - 19 unidades

F \ V	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	14	13	16	15	6	5	12	11	4	3	10	9	2	1	8	7	19	18	17
2	1	17	18	2	6	16	12	4	7	11	14	8	10	13	15	9	19	3	5
3	19	18	17	10	4	11	12	5	6	13	14	7	8	15	16	9	3	1	2
4	6	3	2	1	18	5	8	15	17	4	12	14	16	7	11	13	19	9	10
5	2	1	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
6	6	5	2	1	10	9	8	7	14	13	12	11	19	18	17	16	15	3	4
7	5	1	3	2	9	7	6	8	14	12	11	13	18	17	16	19	4	10	15
8	1	2	3	4	16	15	14	13	8	6	7	5	12	11	10	9	19	17	18
9	2	3	4	1	11	17	16	10	9	15	14	8	6	13	12	5	19	18	7
10	4	3	2	1	8	7	6	5	14	13	12	11	18	17	16	15	19	9	10
11	3	1	2	4	16	15	14	13	12	11	10	9	5	6	7	8	19	18	17
12	19	3	2	1	10	14	15	11	9	12	13	8	7	16	17	6	18	5	4
13	19	14	5	4	18	13	9	8	17	15	10	7	16	12	11	6	3	2	1
14	1	4	3	2	6	8	7	5	10	17	18	9	14	15	16	13	19	12	11
15	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
16	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
17	16	17	18	19	5	11	12	6	3	9	10	4	1	7	8	2	15	14	13
18	4	2	3	1	15	18	10	6	14	17	11	5	13	16	12	7	19	8	9
19	2	4	1	3	7	6	11	5	14	13	15	12	16	19	17	18	9	8	10

Caso K - 4 unidades

F \ V	1	2	3	4
1	1	3	4	2
2	3	2	1	4
3	2	1	4	3
4	3	4	2	1

Caso L - 13 unidades

F \ V	1	2	3	4	5	6	7	8	9	10	11	12	13
1	2	4	8	6	9	5	7	3	13	1	12	11	10
2	12	11	10	9	8	7	6	5	2	1	3	4	13
3	6	8	10	12	13	11	9	7	2	1	3	5	4
4	10	9	8	7	3	5	2	6	1	4	13	12	11
5	3	4	5	6	7	8	9	10	1	2	11	12	13
6	3	4	5	6	7	8	9	10	1	2	13	12	11
7	13	12	7	6	5	4	3	8	2	1	9	11	10
8	12	11	10	9	8	7	6	5	2	1	3	4	13
9	5	6	7	9	8	10	4	3	2	1	11	13	12
10	3	7	8	9	10	11	12	13	1	2	6	5	4
11	10	9	8	5	7	4	6	3	2	1	11	12	13
12	12	11	9	7	6	5	4	3	2	1	8	10	13
13	6	7	8	9	10	11	12	13	1	2	3	4	5

Caso M - 29 unidades

F \ V	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	
1	28	12	13	15	17	20	22	24	27	29	25	23	21	19	16
	14	11	26	10	4	3	8	7	6	9	2	1	5	18	
2	14	13	15	29	19	20	21	22	23	27	26	25	24	28	17
	16	12	18	3	9	10	4	5	6	7	8	11	2	1	
3	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
4	16	17	18	14	20	21	22	23	15	19	13	12	11	10	9
	8	7	3	1	2	24	4	5	6	25	26	27	28	29	
5	5	6	7	8	9	15	16	20	21	22	25	26	23	19	4
	3	2	1	13	14	24	28	29	27	18	17	12	11	10	
6	8	9	10	11	14	27	26	25	22	21	20	16	12	7	4
	3	1	2	5	6	13	15	18	19	29	28	24	23	17	
7	5	11	12	17	18	19	24	25	26	27	29	28	15	14	13
	10	6	4	1	2	3	7	8	9	16	20	21	22	23	
8	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
	14	13	12	11	10	9	8	7	6	5	4	3	1	2	
9	29	12	13	14	15	16	17	18	19	20	11	10	9	8	7
	6	5	21	22	1	2	3	4	23	24	25	26	27	28	
10	3	4	8	14	15	21	26	27	29	28	25	24	23	22	11
	9	7	2	5	12	20	19	18	17	16	13	10	6	1	
11	3	4	7	8	9	25	24	23	22	21	20	19	18	17	13
	6	5	2	1	14	15	16	27	26	28	29	10	11	12	
12	1	2	3	4	23	22	21	20	19	18	17	16	15	14	13
	12	11	10	24	5	6	7	8	9	29	28	27	26	25	
13	1	2	8	9	28	29	10	11	12	16	17	18	19	20	21
	22	4	3	5	26	15	14	27	23	24	25	13	7	6	
14	16	17	18	19	20	21	22	23	24	25	26	27	28	29	8
	7	6	5	4	3	2	1	14	15	10	11	12	13	9	
15	5	6	7	8	9	18	17	16	15	14	13	12	11	10	4
	3	2	1	19	20	21	22	23	24	25	26	27	28	29	

16	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12
	11	10	9	1	3	4	5	8	2	29	28	27	6	7	
17	29	1	4	16	15	5	6	7	8	9	10	11	12	13	14
	2	3	17	18	19	20	21	22	23	24	25	26	27	28	
18	2	1	3	4	5	17	18	22	23	24	25	26	27	13	12
	11	10	9	14	15	16	29	28	21	20	8	7	6	19	
19	13	14	15	20	21	22	25	26	27	28	17	29	12	11	10
	9	8	7	1	2	3	4	5	6	18	19	24	23	16	
20	13	14	15	17	20	23	25	27	29	28	26	24	22	21	19
	18	16	12	2	1	3	4	6	7	8	9	10	11	5	
21	11	12	13	26	27	14	15	28	29	19	18	17	16	25	24
	10	9	8	3	22	23	4	7	6	5	21	20	2	1	
22	2	3	4	6	7	5	8	9	10	11	12	13	14	15	16
	17	18	1	19	20	21	22	23	24	25	26	27	28	29	
23	4	5	6	7	8	25	26	19	10	11	12	13	14	15	16
	2	1	17	18	3	9	20	29	21	28	27	22	23	24	
24	1	6	7	8	9	21	22	23	24	25	20	19	18	15	14
	10	5	4	2	3	11	12	13	16	17	26	27	28	29	
25	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
26	12	13	19	20	21	29	28	27	26	25	24	23	22	18	17
	16	15	14	3	4	11	10	9	8	7	6	5	2	1	
27	12	13	29	14	15	16	17	18	19	20	21	22	23	24	25
	26	27	28	3	4	7	8	9	10	11	6	5	2	1	
28	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11
	10	9	8	26	1	2	3	4	5	6	7	27	28	29	
29	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15
	14	13	12	1	2	3	4	5	6	7	8	9	10	11	

A.3. Script para elaboración de los Frente de Pareto

Script en Elixir

```
defmodule MTAV do
  @root_path "/..."
  @solver "org.gecode.gecode@6.3.0"

  def pareto_diagram(percentage, optimal, coop, draw) do
    model_file_path =
       @root_path <> "/modelos/matrices/max-dev-std.mzn"
    data_file_path =
       @root_path <> "/casos/Minizinc/matrices/coops/#{coop}/#{draw}/fairness.dzn"

    landa = round(optimal * percentage)
    delta = (optimal - landa)..(optimal + landa)

    for s <- delta do
      IO.inspect("==== Running now: #{s} as optimal ====")

      # Change data file to new S as sgo
      :ok = update_data_file(data_file_path, s)

      output_file_path = @root_path <> "/pareto_#{s}_as_sgo.txt"

      command = "
                minizinc --solver #{@solver} --json-stream -s -i --output-time
                --time-limit 2000000 -o #{output_file_path} #{model_file_path}
                #{data_file_path}
              "

      task = Task.async(fn -> System.shell(command) end)
      {_output, exit_code} = Task.await(task, :infinity)

      IO.inspect(exit_code)
    end
  end

  def update_data_file(path, sgo) do
    ...
    ...
  end
end
```

Listing A.1 Codigo Fuente - Elixir - Frente Pareto