UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA DE MINAS

DISEÑO ÓPTIMO DE MINERÍA SUBTERRÁNEA AUTO-SOPORTADA

TESIS PARA OPTAR AL GRADO DE MAGISTER EN MINERÍA

MEMORIA PARA OPTAR EL TÍTULO DE INGENIERA CIVIL DE MINAS

MARÍA CAMILA GRIGALIUNAS GOMPERTZ

PROFESOR GUÍA:
JUAN LUIS YARMUCH GUZMÁN

MIEMBROS DE LA COMISIÓN:
XAVIER EMERY
JOSÉ SAAVEDRA ROSAS
MARCOS GOYCOOLEA GUZMÁN

SANTIAGO DE CHILE

2016

RESUMEN DE LA TESIS PARA OPTAR AL TÍTULO
DE: Ingeniera Civil de Minas y grado de Magister en Minería
POR: María Camila Grigaliunas Gompertz
FECHA: Diciembre 2016
PROFESOR GUÍA: Juan Luis Yarmuch

DISEÑO ÓPTIMO DE MINERÍA SUBTERRÁNEA AUTO-SOPORTADA

La presente investigación trata la obtención del diseño óptimo en minería subterránea auto-soportada, específicamente Sublevel Stoping (SLS) sin relleno. Se tiene como objetivo general la generación e implementación de un modelo de optimización que permita la obtención del layout óptimo de caserones. Esto, con el fin de maximizar el beneficio asociado a la extracción en cualquier proyecto a ser explotado por este método.

Principalmente, se generó un modelo de optimización binaria en el cual se tiene como función objetivo la maximización del beneficio de los caserones a extraer. Asimismo, se incorporaron restricciones técnico-operacionales que corresponden a la imposición de no traslape de caserones, y las restricciones relacionadas con la incorporación de pilares y losas. Con el fin de comparar los resultados, se utilizó el modelo heurístico de diseño de caserones del autor D.S.S. Sandanayake, el cual genera una familia de sets únicos de caserones que cumplen con las restricciones impuestas anteriormente, incorporando una restricción al número de soluciones (Parámetro de la heurística) para la implementación computacional factible del modelo.

Para estudiar lo anterior, se generaron 5 casos de estudio asociados a distintas cantidades de caserones factibles como input. Cada uno consistió a su vez en 10 modelos variantes en ley media, lo que se traduce en un total de 200 corridas; 150 asociadas a tres parámetros de la heurística distintos. Las restantes 50 se corrieron por medio del modelo lineal. Es de resaltar que estas realizaciones se generaron para archivos de caserones factibles con niveles predefinidos, por lo que sólo se activan las restricciones de traslape y pilares contiguos. Asimismo, se generó un caso de estudio general, el cual consistió en un modelo de caserones que se corrió de forma similar a los casos anteriores; sin embargo, se incorporó una corrida asociada al archivo de caserones sin predefinición de niveles para el modelo de Envolvente Económica (modelo lineal).

Los resultados reflejan la eficacia del modelo lineal ya que permite asegurar la optimalidad de los resultados, y es posible correrlo en tiempos razonables. Se tiene que el modelo lineal con niveles genera un resultado 17.17% mayor en comparación al modelo heurístico, con tiempos de ejecución razonables (segundos-minutos en comparación a horas con el modelo de Sandanayake). A su vez, el modelo lineal libre entrega resultados 4.70% mejores que el modelo restringido a niveles; pero a un costo computacional mayor (66 [hrs] extra).

AGRADECIMIENTOS

Quiero darle muchas gracias a mi familia por su apoyo y su amor incondicional. En especial a mi mami, papi y a mis hermanitos, los amo con todo mi corazón y me hacen mucha falta. A pesar de que no estén en este momento presentes físicamente, siempre están conmigo donde quiera que voy. También quiero agradecer a mis abuelitos, Tesoro y Tata, pues sin ustedes todo esto no hubiese sido posible, me han ayudado de variadas formas y siempre han estado para mí durante mi tiempo acá. También muchas gracias a Mochute y Deducas, pues a pesar que nos vemos poco, han sido siempre muy cariñosos conmigo, y parte especial en mi vida. Por último, quiero darle las gracias a mi tía que me recibió durante los años de la universidad.

Agradezco también a mi profesor guía, ¡Gracias por su gran ayuda y su apoyo para que esto pudiera salir adelante! Espero que le vaya excelente en su doctorado, en su vida en general, y en todo lo que se proponga. También quiero dar las gracias al Departamento de Ingeniería de Minas de la Universidad de Chile, por el financiamiento otorgado por la tesis, así como al Centro de Investigación de Operaciones Aplicado a Minería (CIOMIN) del Instituto de Sistemas Complejos de Ingeniería. Por último, a CONICYT por el apoyo del proyecto Conicyt PIA Anillo ACT1407 en la que participan los profesores Emery y Goycoolea. De igual manera, agradezco a cada uno de los profesores que integran la comisión por su ayuda y disposición, les deseo lo mejor en todo.

Gracias también a esas personas especiales que han aparecido en mi vida, ya sea para quedarse o que pasaron por algún tiempo. Todas han formado parte de mí de alguna u otra forma. Me gustaría nombrar algunas personas que son especiales para mí (espero no haber olvidado a alguien): Chere, Jany, Xime Vale, Caro, Vincent, Giovanni, Alvarin, Richard, René, Alan, Flavio, Armando, Oscar, Nico G., Vale B., Japi, Mati, Nachi M., y Pancho R. Quiero dejar claro que el orden en el que fueron nombrados no significa nada jeje. Gracias por haber estado de alguna u otra manera en mi vida, los quiero mucho y espero que sean muy felices siempre.

Durante los años de la U estuve en varias cosillas metida, por lo que quiero también agradecer a todas esas personas que fueron parte de eso. En específico a los del CAM 2013, Feria Empresarial y Difusión, cariños para todos. También quiero agradecer de forma muy especial a Nico, Ingrid, Vero, Javier, Lili, Marce, Paula, Bárbara, Juan, Carlitos y Luchito. Gracias por toda su ayuda, y por considerarme para tantas cosas. Los quiero mucho y espero que nos volvamos a ver de nuevo, han sido muy especiales para mí. De igual forma, me gustaría agradecer a la Mane y a Pili por todo su apoyo, las quiero mucho también.

Finalmente, muchas gracias a todas aquellas personas que han sido de alguna u otra forma de ayuda, o que han estado durante estos años presente. Imposible nombrarlos a todos, pero les deseo siempre lo mejor.

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE ECUACIONES

1. INTRODUCCIÓN

Posterior a la selección del método de explotación para la extracción del mineral de interés, se requiere la realización de la etapa de diseño cuyo objetivo principal es la obtención de un layout de unidades de extracción que maximicen el beneficio del proyecto, de manera que sea posible pasar a la etapa de planificación, y así se pueda generar la extracción de las reservas en el tiempo, por medio de la maximización del valor del proyecto. Teniendo ello en consideración, es de alta relevancia la etapa de diseño ya que ésta influenciará los resultados de la etapa de planificación minera.

En el caso específico de minería selectiva de Sublevel Stoping (SLS), al día de hoy han existido variados alcances para la obtención del diseño óptimo (Alford,1995; Ataee-Pour (2004), Sens (2011), Sandanayake, Topal, Asad (2015), entre otros). Sin embargo, estos se han centrado en modelos heurísticos que no aseguran el alcance del resultado óptimo, y no en todos se ha considerado la incorporación de restricciones operacionales y geomecánicas en la resolución del problema. Asimismo, muchos de los casos estudiados (Ataee-Pour (2004), Sens (2011), Sandanayake et al. (2015), por ejemplo) se han centrado principalmente en Sublevel Stoping con relleno, y no en el caso de minería selectiva auto-soportada; es decir, con la incorporación de pilares para el sustento de las unidades básicas mineras (UBM).

Considerando que la obtención de una solución óptima en el diseño repercute en las etapas posteriores, y, por ende, en el valor del proyecto, y que además los modelos hasta el día de hoy estudiados no garantizan el alcance del óptimo, es relevante el planteamiento de un modelo matemático de optimización que integre las restricciones necesarias para la obtención de un diseño óptimo, el cual en el caso a estudiar corresponde a la distribución de los caserones (UBM) en el layout de explotación. Para ello, se busca entonces un modelo que permita resolver lo anterior, asegurando la optimalidad de la solución, y de forma eficiente computacionalmente.

Actualmente, con los avances computacionales de los últimos tiempos, es posible llegar a plantear y resolver problemas de optimización en tiempos que antes no eran factibles o razonables. Esto, mediante la implementación de códigos computacionales eficientes, y la utilización de diversas herramientas y/o softwares disponibles. Es por ello que a continuación se propone un modelo matemático-computacional que permita la obtención del layout óptimo de caserones sujeto a restricciones que le den operatividad y sustento geomecánico a la explotación.

1.1. Modelo propuesto

Se propone la generación e implementación de un modelo de optimización lineal entero que permita la obtención del layout óptimo en minería subterránea auto-soportada. La unidad básica minera de extracción (UBM) corresponde al caserón, el cual constituye una excavación que permite la extracción del mineral de interés económico.

A partir del modelo de bloques[1] de un respectivo depósito minero, el cual contiene la información geológica requerida para la realización de la etapa de diseño, es posible obtener un modelo económico de bloques en el cual se asocia a cada bloque su valor económico respectivo. Para el cálculo de la utilidad (UT) de cada bloque, se requiere conocer parámetros como el precio del metal de interés, costo mina, costo planta, y recuperación. En la Ecuación 1 se indica cómo se calcula el beneficio asociado a cada bloque.

$$UT\ [US\$] = \left(P\left[\frac{US\$}{lb}\right] - C_v\left[\frac{US\$}{lb}\right]\right) \times L[\%] \times R[\%] \times T[ton] - T[ton] \times C_p)$$

Ecuación 1: Utilidad bloque

Donde,

P: Precio del metal [US$/lb]
C_v: Costo de venta (fundición y refinería) [US$/lb]
L: Ley [%]
R: Recuperación [%]
T: Tonelaje bloque [ton]
C_p: Costo de producción (Costo mina + Costo planta)

Dado que la UBM en minería auto-soportada de Sublevel Stoping corresponde al caserón, es posible adquirir un modelo de caserones factibles a partir del modelo económico de bloques. Lo anterior se realiza por medio de la agregación de bloques contiguos para la generación de un caserón de determinadas dimensiones (largo [m] x ancho [m] x alto [m]), y con un cierto valor económico (correspondiente a la suma del valor de cada bloque asociado). Así, el total de caserones factibles, input para el modelo de optimización, corresponde al caserón generado para cada punto/bloque del espacio que conforma un depósito mineral (Ilustración 1). En particular, la generación de los caserones comienza con el bloque con coordenada mínima en todas las direcciones.

[1] Discretización del yacimiento en bloques de dimensiones determinadas, y con características básicas como la ley mineral, y tonelaje asociado.

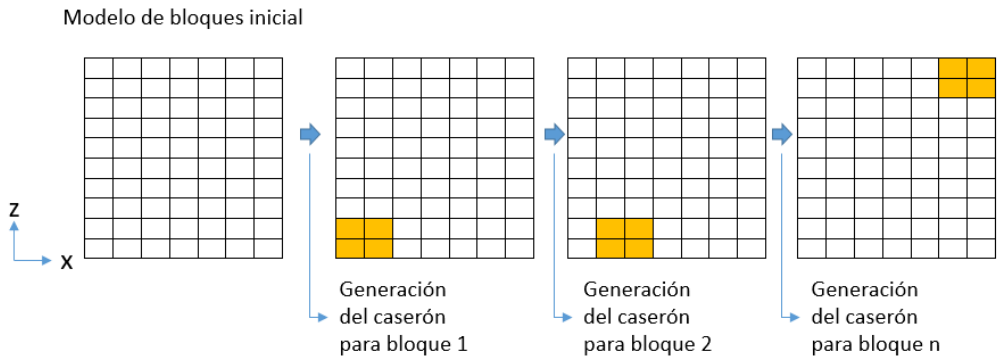


Ilustración 1: Generación caserones factibles para la optimización

Con el fin de entregarle sustento geomecánico a la explotación, es necesario dejar roca entre caserones vecinos. Estos se llaman de diversa forma dependiendo de la función que cumplen. Para caserones que se encuentran en la misma cota (z), y que están ubicados de forma contigua, es requerido disponer de *pilares* que entregan estabilidad a la excavación. Por su parte, para caserones vecinos, pero en cotas distintas, se dispone de *Crown pillars* o *losas* los que entregan estabilidad al método de explotación.

Teniendo en consideración lo anterior, y el hecho de que el modelo de caserones factibles presenta traslapes entre los caserones (los vértices o paredes de los caserones se topan en algún punto del espacio), para la obtención de una solución factible al problema de optimización planteado, es necesario establecer las siguientes condiciones:

1. Los caserones deben cumplir con no traslapar con otros caserones.
2. Entre caserones contiguos del mismo nivel, debe existir un pilar que los separe.
3. Entre caserones vecinos, de distinto nivel, debe existir un Crown Pillar (o losa) que los separe.

Con ello, es posible la optimización del problema, considerando como función objetivo la maximización del beneficio total de extracción de los caserones. Para ello, se propone en la implementación de un modelo de optimización lineal entera, que, mediante las restricciones impuestas, seleccione los caserones a ser extraídos, y que maximice el beneficio de la extracción. De esta forma, se obtiene la envolvente económica óptima.

Para comparar los resultados obtenidos mediante la presente investigación, se propone la utilización de la heurística de D.S.S. Sandanayake (2014), la cual fue ideada para la obtención del layout óptimo de caserones a ser explotados por medio de Sublevel Stoping con relleno, y que se explica en las siguientes secciones. Dado que el modelo propuesto se encuentra asociado a minería auto-soportada (sin el uso de relleno), y se busca comparar la eficiencia en términos de beneficio económico y tiempos de corrida, se propone la generación de caserones posibles (o factibles) de la siguiente manera:

1. Generando caserones posibles con intervalos de losa predefinida. A este se le llamará modelo con *Niveles predefinidos.*
2. Generando caserones posibles para cada bloque del modelo de bloques, sin losa predefinida. A este se le llamará modelo *sin niveles predefinidos* o *libre.*

A continuación, se presentan los alcances y objetivos de la presente investigación.

1.2. Alcances

Los alcances que acotan el presente trabajo se resumen en los siguientes puntos:

- La investigación se encuentra enfocada en la generación de un modelo de optimización del layout óptimo de caserones para minería subterránea auto-soportada de Sublevel Stoping.
- El modelo se propone para yacimientos a ser explotados por medio de Sublevel Stoping sin relleno.
- El modelo de optimización tendrá un enfoque estratégico, es decir, con horizonte de evaluación de largo plazo en términos del diseño y planificación minera.
- El modelo incorporará como restricciones de tipo geo-mecánico, restricciones operacionales asociadas al tamaño mínimo y máximo de caserones, pilares entre caserones del mismo nivel, y pilares Crown Pillar para caserones ubicados en distintas cotas.

1.3. Objetivo General

Generar e implementar un modelo de optimización para abordar la problemática de diseño y obtención del layout óptimo en minería subterránea auto-soportada; en particular Sublevel Stoping (SLS). Mediante la incorporación de restricciones técnico-económicas y geo-mecánicas, con el fin de obtener los caserones que maximicen el beneficio total de extracción.

1.4. Objetivos Específicos

- Implementar el modelo optimización heurística de D.S.S Sandanayake para el diseño de caserones.
- Generar un modelo de optimización que permita resolver el problema de la envolvente óptima del diseño en minería selectiva auto-soportada.
- Incorporar restricciones que permitan asegurar el cumplimiento de restricciones operacionales y geo-mecánicas para la resolución del problema de optimización.

2. ANTECEDENTES

El capítulo a continuación contiene todos los antecedentes bibliográficos que son fundamentales para el entendimiento y desarrollo del trabajo de investigación en cuestión. Principalmente, contempla conceptos como minería subterránea selectiva del tipo auto-soportada, algoritmos para obtención del diseño, y herramientas de optimización.

2.1. Minería subterránea selectiva: Sublevel Stoping

El proceso minero de extracción de mineral y que es desarrollado bajo superficie corresponde a minería subterránea. Los métodos de explotación desarrollados de esta forma se clasifican en tres: Auto-soportados, Artificialmente soportados y de Caving. El primero contempla la minería subterránea selectiva con métodos como Room & Pillar y Sublevel Stoping, los cuales al final de la extracción de la unidad básica minera consisten en cámaras abiertas, y en los cuales se busca la extracción de mineral principalmente, y la menor cantidad de material estéril. Por su parte, los métodos artificialmente soportados contemplan el uso de relleno para la sostenibilidad de la unidad básica minera posterior a la extracción, dentro de los cuáles se encuentran el Cut & Fill y Bench & Fill, siendo también de carácter selectivo. Finalmente, los métodos de caving consisten en métodos de hundimiento del mineral para yacimientos y explotaciones de carácter masivo. En el último se encuentran métodos como Block/Panel Caving y Sublevel Caving.

La selección de un método de explotación sobre otro tiene relación con el tamaño, orientación y características del cuerpo mineralizado. En el caso particular de Sublevel Stoping, el cual se describirá con mayor detalle para efectos de la presente investigación, se tiene la extracción del mineral en yacimientos de carácter masivo o tabular, generalmente sub-verticales, donde se presenta una buena competencia de roca caja y mineral (Villaescusa,2014).

Asimismo, la unidad básica de explotación corresponde al caserón (Ilustración 2) el cual consiste en una excavación realizada por medio de perforación y tronadura secuencial para conseguir el quiebre de la roca, y su posterior extracción por medio de embudos recolectores en su base. De esta forma, se realiza la extracción del mineral por medio de los caserones en distintos niveles, dispuestos de forma tal de maximizar el valor económico.

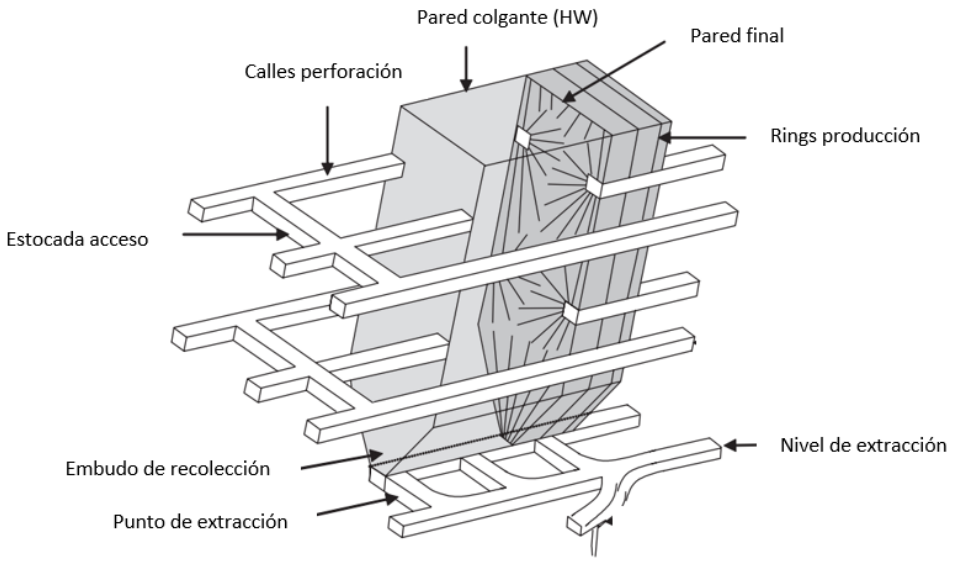


Ilustración 2: Vista 3D Caserón Sublevel Stoping. (Fuente: Villaescursa, E., Geotechnical Design for Sublevel Open Stoping., CRC Press, Taylor & Francis Group., pg. 6.,2014)

Dado que el caserón es dejado como cámara abierta al final de la extracción del mineral, es requerido que el diseño de la unidad de explotación considere parámetros geomecánicos que le otorguen factibilidad y seguridad a la explotación. Además, se requiere disponer de pilares entre caserones contiguos y entre niveles de explotación, los cuales permitan el soporte de la excavación y el método, respectivamente.

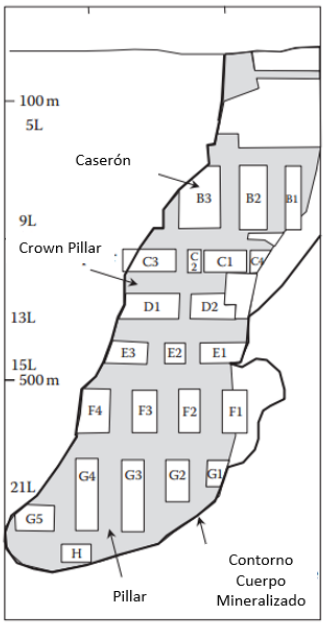


Ilustración 3: Perfil Explotación Sublevel Stoping y Pilares (Fuente: Villaescursa, E., Geotechnical Design for Sublevel Open Stoping., CRC Press, Taylor & Francis Group.,pg. 55., 2014)

Como se mencionó con anterioridad, para el sustento geomecánico de la excavación (UBM) en minería subterránea auto-soportada, es necesario dejar roca entre caserones contiguos de una misma cota, constituyendo de esta forma un *pilar* (Ilustración 3). Por su parte, entre niveles de extracción, es requerido dejar el macizo rocoso intacto, de manera que sea posible asegurar la estabilidad del método de explotación, así como el ingreso para la extracción del mineral de interés. De esta forma, la roca entre caserones de distinta cota, constituye el *Crown pillar* o *losa*.

2.2. Algoritmos de diseño y selección de caserones

Dentro de la etapa de diseño de minería subterránea auto-soportada de Sublevel Stoping el aspecto más importante a considerar es la optimización de la ubicación de los caserones y su delimitación. En la práctica actual, los caserones son generalmente diseñados siguiendo un patrón base y dimensiones, y los límites se deciden usando reglas básicas de dedo. También, en muchos casos los caserones se sitúan en ubicaciones con una mínima ley de corte. Sin embargo, esta práctica no necesariamente produce una utilidad óptima o el mejor Valor actual neto (Sens, 2011). Es así, como la selección de un layout de caserones óptimo es el núcleo para el proceso de planificación minera en la mayoría de los métodos de explotación subterráneos, ya que dictamina la maximización del Valor actual neto en el subsiguiente agendamiento de producción (Sandanayake et al., 2015).

Alford, en 1995, desarrolló el algoritmo *Floating Stope Algorithm* (Alford, 1995) en el cual primero se especifica una ley de corte para separar bloques de mineral y lastre, así como una ley mínima para los caserones. De esta forma, se procede a flotar un bloque rectangular, que representa el tamaño mínimo de caserón, por el modelo de bloques de forma de identificar la posición de mayor ley para la ubicación del caserón. Básicamente corresponde a un acercamiento heurístico, en el que el problema principal es determinar si cualquier bloque que cumple la restricción de estar sobre la ley de corte especificada, puede estar incluido en un caserón que cumple con la ley requerida para el ingreso a planta. En el caso en que existan muchas alternativas de caserón para un bloque, se determina que este pertenece al caserón con la ley más alta. De esta forma, el output del algoritmo corresponderá a un modelo de bloques que define el volumen de la envolvente óptima del caserón, así como su mejor ley, siendo ésta el borde en el cual el planificador minero puede determinar la ubicación óptima.

Es un modelo simple, lo cual es una ventaja para su utilización. Sin embargo, dentro de los inconvenientes principales se tiene que el diseño y ubicación final de caserones es decidido por un ingeniero con experiencia que tome consideraciones prácticas, por lo que el algoritmo no determina el caserón óptimo pues sólo delimita su ubicación con envolventes, sin entregar una ubicación exacta.

Por otro lado, se encuentra el *Multiple Pass Floating Stope* Algorithm *(Cawrse, 2001)*, el cual está basado en los mismos principios que el *Floating Stope Algorithm* para delinear las zonas económicas según la ley de cabeza del caserón y la ley de corte. Sin embargo, este algoritmo genera varias envolventes diferentes basadas en distintos parámetros de entrada, con lo que se puede utilizar para seleccionar y diseñar caserones. Al igual que el modelo anterior, no se garantizan resultados óptimos y se requiere la decisión final de un ingeniero.

Otro algoritmo desarrollado corresponde al *Maximum Neighborhood Value (Ataee-Pour, 2004).* Básicamente consiste en que un caserón se ubica en una esquina del modelo de bloques y se calcula su valor económico. De ser positivo, el caserón se mantiene y los bloques considerados dentro de éste, son etiquetados de forma que no se tomen en cuenta nuevamente. Por otro lado, si el valor económico es negativo, el caserón se descarta, y el procedimiento se repite desplazando el set de bloques. De esta forma se evalúa sistemáticamente el modelo de bloques, y se identifica la mejor combinación de bloques evaluando todas las posibles vecindades (alrededor de cada bloque). Los tamaños de caserón se encuentran principalmente restringidos por las propiedades geomecánicas del cuerpo mineralizado y de la roca circundante, así como los tamaños de los equipos utilizados durante el proceso de minado. El tamaño mínimo debe ser diseñado de forma tal que sea suficiente el espacio provisto para las actividades de perforación, tronadura y carguío de marina[2], mientras que el límite máximo de dimensiones de un caserón está usualmente dictado por los factores geotécnicos para impedir colapso de paredes.

Dentro de las desventajas principales que conlleva este algoritmo se encuentra el hecho que la solución puede variar dependiendo del punto de partida, por lo cual se generan diversos layouts de un mismo cuerpo mineralizado. También, independiente de que todos los caserones seleccionados por el algoritmo presenten valor económico positivo, no necesariamente la combinación de los caserones puede ser la combinación óptima de todos los caserones posibles. A esto se le puede sumar el hecho que los bloques que son examinados primero en el proceso son seleccionados de forma preferencial. Por último, el modelo desarrollado no considera costos asociados al tamaño de caserón.

Sens, en el 2011, desarrolló un algoritmo de optimización combinatorial que delimita el mejor set de caserones para maximizar la utilidad basado en la selección de caserones que cumplan con ciertos criterios especificados por el usuario (Sens, 2011). Entre estos se encuentra la maximización de la utilidad del caserón (entregando el diseño minero que maximiza el beneficio global posible), maximización de la utilidad del caserón por metro cuadrado (entregando el diseño minero con mayor VAN en relación con el criterio anterior), y, por último, maximización de la utilidad del caserón dividido por su tiempo total de minado (resultando en el mejor VAN puesto que los mayores ingresos se obtienen mientras se tiene una menor tasa de descuento).

[2] Producto que resulta después de la tronadura, y que puede ser mineral o estéril.

El procedimiento para la optimización de los caserones inicia con determinar la matriz de valores económicos de bloques, y establecer los distintos tamaños de caserón en todas las ubicaciones dentro del modelo de bloques. Lo anterior es realizado por el usuario. Luego, se descartan todos los caserones con utilidades negativas de forma que con el algoritmo se pueda escoger los caserones. Así, se calculan los valores económicos de los caserones y aquél con mayor beneficio es seleccionado, evaluando dentro de un radio seleccionado por el usuario las diversas combinaciones posibles. Aquellos bloques escogidos se etiquetan para que no estén disponibles para selección nuevamente, y así el algoritmo continúa hasta que ya no existan más caserones determinados por el usuario. Es de resaltar que, dependiendo del radio de optimización escogido, las regiones serán mayores y el resultado estará más cerca del verdadero óptimo, ya que básicamente se busca la mejor combinación de caserones dentro del radio determinado y para el caserón con mayor utilidad.

Con el fin de obtener el máximo VAN, Sens realizó un modelo que le permitiera secuenciar los caserones. Básicamente produce un programa que contiene el número, la producción diaria, la fase de producción y el flujo de caja de los caserones.

De forma similar, Little, Knights y Topal (2013) formularon un modelo de optimización integrado usando técnicas de programación entera para generar planes de largo plazo enfocados en optimizar layouts de caserones y programas de producción para operaciones de Sublevel Stoping. El modelo primero convierte todos los bloques del modelo a un tamaño regular para facilidades en el procesamiento de los datos. Con ello, se procede a realizar la optimización considerando dos tamaños de caserones fijos, y establecidos por el usuario. Básicamente con el input anterior, calcula el total de caserones posibles que existen dentro de los límites del modelo de bloques, y según un intervalo regular en cada dirección (X, Y e Z) crea caserones a evaluar con el modelo. Finalmente, se identifican aquellos caserones con utilidad con base en la resolución del modelo de planificación integrada, y refiriéndose a consideraciones geotécnicas cómo la determinación de caserones adyacentes, caserones que deben encontrarse distanciados entre sí y niveles de extracción.

Recientemente, en 2014, Sandanayake presentó su tesis de doctorado, en el cual presenta un acercamiento heurístico para el diseño óptimo del layout de una mina explotada de forma subterránea. En función de las restricciones físicas y geotécnicas (o las características del material disponible), se definen tamaños de caserones para así maximizar el valor económico, con base en un modelo de bloques como input. Se estudió el desempeño del algoritmo para dos casos donde se evaluaron tamaños fijos de caserones en las tres dimensiones, sin y con la incorporación de pilares. De igual forma, se estudiaron otros dos casos en los cuales se consideran largos variables de caserón en los tres ejes coordenados; es decir, el número de bloques a considerar en cada eje no es necesariamente el mismo. El último, también se realizó sin y con la incorporación de pilares.

El algoritmo se desarrolla primero convirtiendo el modelo de bloques en uno económico, para de esta forma agregar bloques que permitan generar un set de posibles caserones, y extraer todos con beneficio económico mayor a cero. Posterior a ello, se crea una familia de sets de caserones que cumplan con la condición de no traslape, es decir, caserones en donde los vértices o paredes de la excavación no coinciden o se tocan. Con lo anterior, se procede a calcular el valor económico de cada set, para determinar aquél de mayor beneficio, el cual corresponde al óptimo de la solución.

Dado que en el modelo anterior existen infinitas combinaciones para generar sets de caserones que no se traslapan, lo que implica una alta complejidad computacional del problema, los autores limitaron los sets de caserones que no comparten bloques en común, a un número limitado de iteraciones. Asimismo, una desventaja del presente acercamiento heurístico consiste en que está limitado a tamaños de caserones pre-definidos.

La implementación del modelo de Sandanayake (2014), generó 6 casos que varían según si se generaban caserones de tamaño fijo o variable y si se adicionaban pilares y niveles, o no. La generación de la familia óptima de sets de caserones para estos casos se obtenía en un promedio de 5 horas (para un modelo de 287,984 bloques de 10[m]x10[m]x10[m]), donde casos se resolvían de forma muy rápida (1 hora) y otros requerían de mayor tiempo computacional (14 horas). Su modelo fue validado con el modelo de Sens (2011), mencionado con anterioridad, así como el modelo de Ataee-Pour (2000), con el *Maximum Value Neighborhood.*

En la Tabla 1 se presenta un resumen de los algoritmos desarrollados para el diseño en minería de auto-soportada.

Por otro lado, Little et al. (2015) genera un modelo de planificación integrada en el cual los layouts de caserones y su secuenciamiento se optimizan de forma simultánea, incorporando restricciones que limiten la producción en caserones vecinos, la capacidad mina, el relleno y producción entre niveles, entre otros. Asimismo, se incorporaron precedencias en términos de que primero debe ir un periodo de preparación, seguido por dos periodos de extracción, un periodo vacío, y, por último, un periodo de relleno. Los resultados fueron comparados con la resolución del diseño y planificación de forma tradicional. Dentro de los resultados obtenidos se encontró que el primer enfoque entrega al final de la vida mina un VAN 36% mayor, lo que corresponde a una metodología en la que es necesario ahondar para la obtención de proyectos más realistas y óptimos.

Tabla 1: Resumen algoritmos diseño y selección caserones

Resumen algoritmos diseño caserones		
Algoritmo	**Ventajas**	**Desventajas**
Floating Stope Algorithm-FSA, Alford (1995)	Simple	Diseño y ubicación final decidido por ingeniero (no optimalidad)
		Caserones traslapados
		Tamaño fijo caserón
		No considera pilares
Multiple Pass Floating Stopes Algorithm-MPFSA, Cawrse (2001)	Simple	Diseño y ubicación final decidido por ingeniero (no optimalidad)
	Considera mayor cantidad de parámetros de entrada en relación a FSA	Caserones traslapados
		Tamaño fijo caserón
		No considera pilares
Maximum Value Neighborhood-MVN, Ataee-Pour (2004)	Tamaños caserones restringidos por propiedades geomecánicas	Solución varía en función del punto de partida por lo que se generan distintos layouts
		Combinación caserones no necesariamente óptima pues bloques examinados primero tienen mayor preferencia
		No considera costos asociados al tamaño de caserón
		No considera restricciones geomecánicas a la pared del caserón
		No considera pilares
Stope Algorithm, Sens (2011)	Tamaño caserón fijo o variable en 3 direcciones	No considera pilares entre caserones contiguos
	Considera crown pillar	No considera crown pillar continuo pues depende de la posición del caserón: problema accesos
	Caserones no traslapados	
Planificación Integrada, Little; Knights; Topal (2013)	Considera geotécnica para determinación de caserones adyacentes	Tamaño caserón fijo y establecido por usuario
	Caserones no traslapados	No considera pilares
		Problema accesos
Acercamiento heurístico para diseño óptimo, Sandanayake (2014), Sandanayake et al., (2015)	Restricciones geotécnicas para definición de tamaños de caserones	Tamaños fijos caserón predefinidos para todo el modelo de bloques
	Incorporación pilares	
	Caserones no traslapados	

2.3. Modelos y herramientas de optimización

Existen variados métodos de optimización los cuales se clasifican en dos principales: los clásicos y los metaheurísticos. Dentro de los primeros se encuentra la optimización lineal, entera, mixta, no lineal, estocástica, entre otros. Para estos, es posible decir que buscan y garantizan un óptimo. Por su parte, los modelos metaheuristicos, tales como los algoritmos evolutivos y búsquedas heurísticas, constituyen mecanismos para alcanzar un óptimo, aunque no garantizan su alcance (Ramos et al. (2010)).

Los problemas de optimización clásicos, en su formulación, incluyen una función objetivo, variables de decisión, y restricciones que definen una región factible. Básicamente, la primera constituye la meta (o criterio) que se desea obtener, y puede ser maximizada o minimizada con el fin de obtener el óptimo del problema. Por su parte, las variables de decisión corresponden, o más bien, representan, las cantidades a ser determinadas. Estas son las decisiones que se pueden tomar para afectar el valor de la función objetivo. En cuanto a las restricciones, éstas constituyen el conjunto de relaciones, expresadas mediante ecuaciones, que ciertas variables están obligadas a satisfacer (Ramos et al. (2010)). De esta forma, es posible obtener el set de soluciones factibles y obtener finalmente una solución óptima.

Cuando se desea modelar situaciones donde es requerido capturar decisiones "Sí/No" como extraer un bloque o no, por ejemplo, los problemas de Programación Entera (IP) son de utilidad. Básicamente, estos son aquellos en los cuales las variables decisión deben tomar valor entero en la solución final, y la función objetivo cumple ser una función lineal, así como las restricciones que definen el problema. En particular, si las variables de decisión sólo toman valores 0 o 1, se tiene un problema de programación binaria. Con lo anterior la representación matemática se presenta, a continuación:

$$Función\ objetivo: Minimizar/Maximizar \quad \sum C_j x_j$$

$$\begin{aligned} &Sujeto\ a: \\ &Restricciones: \\ &\sum_j a_{ij} x_j \leq b_i, \qquad i = 1,2,\dots,M \\ &x_j \geq 0, \qquad j = 1,2,\dots,N \\ &x_j \in \{0,1\} \end{aligned}$$

Ecuación 2: Estructura matemática programación binaria

Donde, c_j, b_i y a_{ij} son coeficientes conocidos, y x_j es la variable desconocida.

3. METODOLOGÍA

A partir de la problemática presentada, se propone la realización de un modelo de optimización entera, el cual incorpore los parámetros de ingreso técnico-económicos y restricciones de tamaño de caserón, económicas y de capacidad al diseño de yacimientos a ser explotados por Sublevel Stoping (SLS) sin relleno.

La metodología a seguir para la resolución del problema consta de tres etapas principales: Formulación del modelo de optimización, Optimización, Validación y Análisis de resultados. En la primera se definen las restricciones que definen la región factible de trabajo, así como la función objetivo que provee un medio para comparar las soluciones dentro de esa región. Con ello, en la etapa de optimización se obtienen realizaciones de lo modelado, para así dar lugar al análisis de los resultados y su comparación con el caso base. Teniendo lo anterior en mente, se presenta a continuación una metodología más detallada para la resolución del problema planteado.

1. Recopilación de antecedentes y aportes a la investigación.
2. Definición de los parámetros de entrada globales del problema. Entre estos se encuentran los parámetros económicos y recuperación de metal.
3. Definición de los parámetros y restricciones operacionales y geomecánicas del problema. Principalmente, la definición de los tamaños mínimos y máximos de caserón.
4. Realización del modelamiento de la etapa de diseño y obtención del layout óptimo de caserones en función de las restricciones operacionales impuestas.
    a. Según el modelo heurístico de D.S.S. Sandanayake.
    b. Según un modelo de optimización entera de envolvente económica
5. Optimización y obtención de resultados con ambos modelos.
6. Análisis de resultados y validación del modelo.

4. ALGORITMO D.S.S. SANDANAYAKE

El algoritmo de optimización con el cual se busca comparar y validar los resultados obtenidos del modelo propuesto (modelo lineal de envolvente óptima) corresponde a la heurística de optimización del diseño de caserones más reciente encontrado en la literatura, y correspondiente a la delimitación de caserones de D.S.S. Sandanayake (2014). La metodología general para el modelo de optimización tradicional se presenta en la Ilustración 4, la cual es llevada a cabo mediante un código en Python y Java.

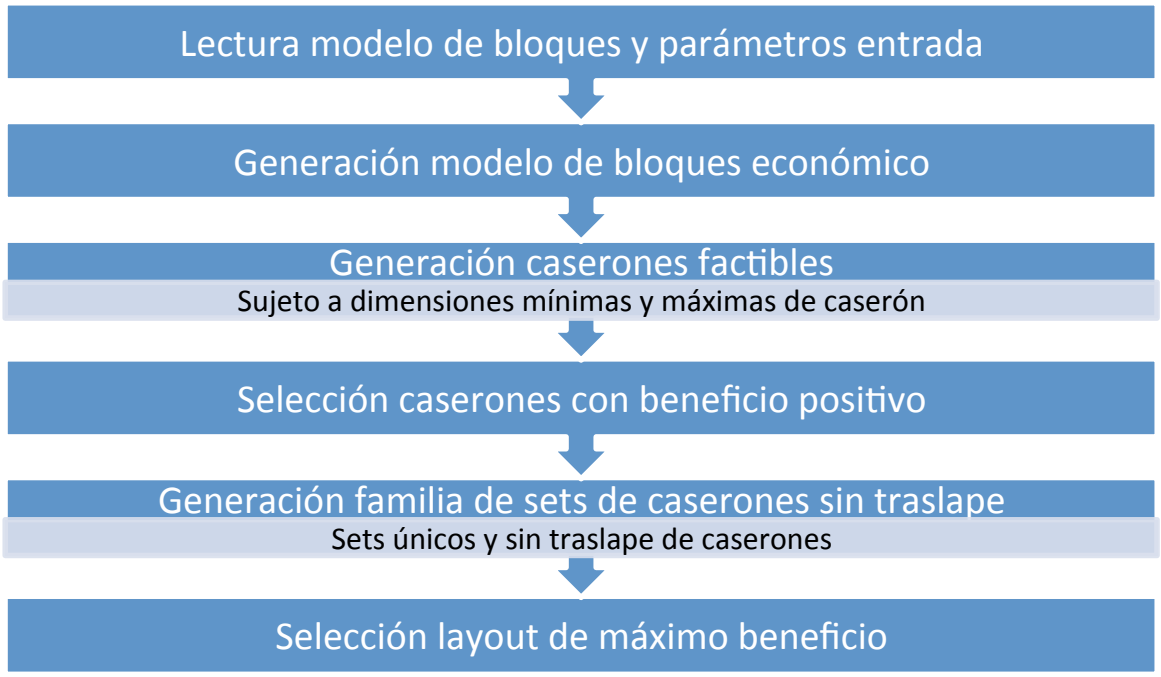


Ilustración 4: Metodología general Algoritmo de Sandanayake (2014)

A continuación, se presenta en detalle lo referente al código en Python y Java.

4.1. Obtención layout óptimo caserones

4.1.1. Generación de caserones-Modelo Python

Como se presentó con anterioridad, para la comparación con respecto al modelo propuesto, se recurrió a la programación de un modelo en Python y Java que permitiese la obtención del layout óptimo de caserones. El modelo escogido para la obtención del layout óptimo (D.S.S. Sandanayake, 2014) corresponde a una heurística que busca obtener el set de caserones sin traslape que maximice el beneficio total por la extracción. Para ello, el modelo de bloques entregado debe ser convertido a un modelo de bloques económico que incorpore el valor de cada bloque según los parámetros de ingreso entregados (sección 1.1). De esta forma es posible obtener el valor de los caserones factibles a generar, considerando un

caserón como la agregación de bloques según restricciones operacionales y geotécnicas dadas por las dimensiones mínimas y máximas que debe tener (Ilustración 1).

Teniendo lo anterior en consideración, la generación de caserones factibles se realiza para cada posición (bloque) del modelo de bloques de forma de obtener todas las posibilidades de unidad básica minera según las restricciones entregadas, entregando la posición del centroide del caserón generado en conjunto con su valor económico. Es de notar que los caserones obtenidos presentan traslape de bloques, razón por la cual es necesaria la etapa de la generación de una familia de sets únicos de caserones que permita obtener el layout de mayor valor. Una vez obtenido el total de caserones factibles, dado que interesa la extracción de mineral y no estéril, se procede a descartar los caserones que presentan beneficio económico negativo.

En la sección 10.1 se explica en detalle el código implementado en el lenguaje Python. El resultado de esta etapa entrega un archivo csv con 9 columnas correspondientes a las coordenadas del origen del caserón en cada dirección (x, y, z), el tamaño del caserón generado (sizex, sizey, sizez), y los parámetros propios como ley (cut), tonelaje (ton) y valor económico. En la Ilustración 5, se presenta un ejemplo de los archivos utilizados y generados para el modelo de caserones factibles. Es de resaltar el hecho que la generación de caserones posibles constituye parte fundamental para el proceso de optimización por medio de la metodología de D.S.S. Sandanayake, así como la metodología propuesta a ser explicada en la sección 5.

Modelo de bloques

cenx	ceny	cenz	cut	ton
10	10	10	0.63	2700
10	20	10	0.72	2700
10	30	10	2.03	2700
10	40	10	0	2700
...	...	...	...	...
50	50	50	0.5	2700

Modelo económico

cenx	ceny	cenz	cut	ton	valor
10	10	10	0.63	2700	33333
10	20	10	0.72	2700	33345
10	30	10	2.03	2700	50000
10	40	10	0	2700	-1534
...	...	...	...	...	...
50	50	50	0.5	2700	25680

Modelo de caserones

cenx	ceny	cenz	sizex	sizey	sizez	cut	ton	valor
10	10	10	30	30	30	0.75	21600	50000
10	20	10	30	30	30	0.85	21600	54689
10	30	10	30	30	30	1.6	21600	70005
10	40	10	30	30	30	0.5	21600	30000
...	...	...	...	...	...	...	...	...
30	50	40	30	30	30	0.7	21600	49980

Ilustración 5: Ejemplo de los archivos para la generación del modelo de caserones

4.1.2. Generación niveles- Sandanayake (2014)

El algoritmo de Sandanayake (2014) incorpora la generación de los niveles de extracción durante la generación de los caserones posibles. Dado que su heurística fue implementada para Sublevel Stoping con relleno, el autor no incorpora Crown pillars puesto que cada caserón es extraído y rellenado posteriormente. Teniendo lo anterior en consideración, en lugar de generar caserones para cada fila/bloque del modelo de ingreso, Sandanayake incorpora la restricción de que la agregación de bloques en el eje vertical sea según la altura máxima del caserón. Es decir, durante el algoritmo de generación se impone la

restricción de buscar caserones factibles con la búsqueda en el eje “z” sujeta a un incremento en bloques equivalente a la altura máxima del caserón (Ilustración 6). Lo anterior, se traduce en un menor número de caserones posibles en el caso de dimensiones fijas.

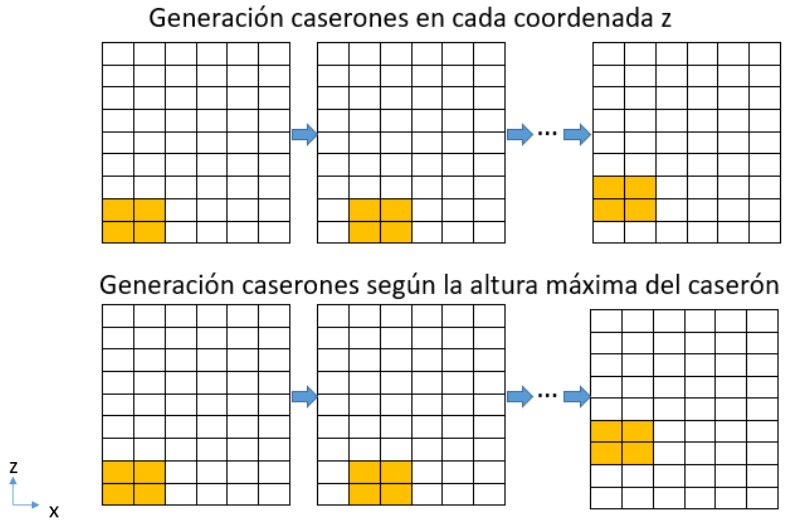


Ilustración 6: Generación caserones factibles

Dado que se analiza el caso de minería subterránea auto-soportada con pilares, y no asociado a relleno como lo plantea el autor Sandanayake, en lugar de generar los caserones factibles mediante una búsqueda en la cota z asociada a la altura máxima del caserón, se propone realizar esto para una distancia que corresponde a la suma de la altura máxima del caserón más la altura del Crown Pillar (Ilustración 7).

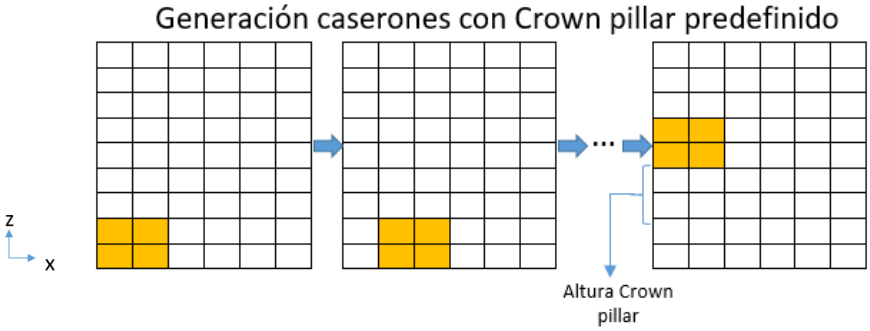


Ilustración 7: Generación caserones factibles con Crown pillar

Asimismo, al predefinir los niveles de caserones es posible sólo llegar a una solución que es óptima para esa determinada configuración, y no para el modelo de bloques o caserones completo. Si bien, el autor del modelo ideó su algoritmo para minería de Sublevel Stoping con relleno -permitiéndole así ubicar caserones encima de otros- dadas las características de su algoritmo, se cree que una

solución como la propuesta por Little et al. (2015) entregaría mejores resultados para tal problema. Dado que la actual investigación está centrada en minería auto-soportada, para trabajar con la heurística de Sandanayake se debe realizar lo mencionado en el párrafo anterior.

4.1.3. Obtención familia set de caserones - Modelo Java

Con el fin de generar la familia de sets únicos de caserones según el algoritmo de Sandanayake se recurrió a la utilización del lenguaje de programación Java, considerando que computacionalmente los tiempos de corrida son inferiores a la programación en Python.

El input para la obtención de lo anterior, corresponde al set de caserones positivos que se obtuvo del modelo en Python. A partir de ello, se crea un conjunto vacío que corresponderá a la familia de sets posibles. Se procede a agregar elementos a éste, según iteraciones sobre los caserones con beneficio positivo. Básicamente, en la medida que se itera sobre cada caserón se verifica que éste no traslape con el anterior y que se cumpla la restricción de pilares. De hacerlo, se descarta y se continúa con el algoritmo. En caso de que no exista traslape, y se cumpla el ancho del pilar entre los caserones, se guarda el subconjunto en la familia, y se continúa con las iteraciones hasta que todos los caserones hayan sido evaluados. La solución óptima corresponde al set de caserones con mayor valor económico. El diagrama de flujo de la tesis doctoral de Sandanayake se presenta en la Ilustración 51, resumiendo la generación de la familia de sets únicos de caserones. Asimismo, a continuación, se muestran los pasos del algoritmo.

1. Lectura de los valores económicos del caserón s (del modelo de caserones positivos: γ_s).
2. Creación de una familia nula de sets de caserones (F_k). Se itera sobre los pasos 3-9 para todo caserón perteneciente al modelo de caserones positivos ($\forall\ s\ \in\ \gamma_s$).
3. Se itera sobre los pasos 4-9 para todo set $\vartheta_k \in F_k$.
4. Se crea una familia nula de sets de caserones T_k.
5. Se crea un set de caserones $\vartheta_{k'}$, y se agrega el caserón s a $\vartheta_{k'}$. Si algún caserón $s' \in \vartheta_{k'}$ no traslapa con s, y cumple con la distancia mínima del ancho del pilar, entonces se procede a combinar los sets $\vartheta_{k'}$ y ϑ_k.
6. Se crea un set de caserones $\vartheta_{k''}$, y se agrega el caserón s a $\vartheta_{k''}$.
7. Los sets $\vartheta_{k'}\ o\ \vartheta_{k''}$ se descartan si son subsets de cualquier set $\vartheta_k\ \in F_k$.
8. Se agrega $\vartheta_k, \vartheta_{k'}, \vartheta_{k''}$ a T_k.
9. Se asigna T_k a F_k, y se itera sobre los pasos 4-8 para todo $\vartheta_k\ \in F_k$.
10. Se remueven todos los sets $\vartheta_k\ \in F_k$ que son subsets de otros sets en F_k para generar F_u.
11. Se determina el valor económico de cada set perteneciente a la familia de sets de caserones F_u, y se escoge aquél con máximo valor como la solución óptima.

Es de resaltar que la cantidad de soluciones asociadas al problema expuesto puede ser infinita. Es por ello que, con el fin de disminuir este tiempo y llegar a una solución razonable acorde a éste, el autor limita las soluciones por medio de un valor que llama *Ctk*. Dado que el algoritmo genera sets de familias de caserones únicas (sin traslape y con distancia del pilar) a partir de iteraciones con los caserones de ingreso, Sandanayake procede a limitar la cantidad de sets de familias con un valor Ctk que parte en cero e incrementa hasta 10,000. Con esto, se pretende evitar que el árbol de búsqueda crezca de forma exponencial en cada nivel de recursión, quedándose con las Ctk mejores soluciones. En cada iteración de caserones, ordena los sets según valores descendentes del beneficio de extracción total para siempre quedarse con los de mejor valor económico.

En la sección 10.2 se explica de manera general el código implementado en Java.

5. MODELO OPTIMIZACIÓN LINEAL

El problema de la obtención del layout que maximiza el beneficio puede ser modelado por medio de un modelo de optimización binaria cuya función objetivo corresponde a la maximización del beneficio asociado a la extracción de los caserones, sujeto a restricciones geométricas pertinentes. Dentro de éstas, se encuentran las restricciones que modelan el problema minero de forma realista, como la incorporación de Crown pillars y pilares que den soporte geo-mecánico a las excavaciones; así como, la selección de caserones que no comparten bloques en común (no traslapan entre sí). A continuación, se presenta el modelo que permite generar la envolvente óptima donde i corresponde a la notación para referirse a los caserones. Es de resaltar que este problema fue programado en el lenguaje Python con la incorporación del optimizador Gurobi. En la sección 10.3 se presenta de manera general, como se implementó el procesamiento de datos (sets) para la optimización.

5.1. Modelo generación envolvente económica de caserones

5.1.1. Sets -Modelo generación envolvente económica

A continuación, se presentan los sets requeridos por el modelo de optimización lineal para la determinación de la envolvente económica óptima de caserones:

N: Set de todos los caserones factibles

bc_i: Set de todos los caserones que comparten bloques en común con el caserón i (traslapados)

cn_i: Set de todos los caserones que no comparten niveles de extracción en común con el caserón i

ca_i: Set de todos los caserones que no cumplen con la distancia mínima requerida por el ancho del pilar con respecto al caserón i

ne_i: Set de todos los caserones que se encuentran en el mismo nivel de extracción al caserón i

Es de resaltar que el set bc_i se obtiene de buscar para cada caserón i aquellos que cumplen con las ecuaciones de traslape presentadas en la Ecuación 11 (sección 10.2), y que permiten determinar si dos caserones se tocan en alguna de sus paredes. Asimismo, el set ca_i se obtiene de verificar que para un plano xy dado (o nivel en z fijo) no se cumpla que la distancia entre dos caserones debe ser de al menos el ancho del pilar, como se presenta a continuación:

$$\left|C_x^i - C_x^{i'}\right| \geq p \ \cup \ \left|C_y^i - C_y^{i'}\right| \geq p, \qquad \forall\, i, i' \in ne_i$$

Ecuación 3: Restricción ancho pilar entre caserones

En relación a la ecuación anterior, se debe tener en cuenta que C_x^i para cualquier caserón i corresponde a la coordenada del centroide. Básicamente, se verifica que entre dos caserones contiguos de una misma cota z, exista una distancia mayor o igual al ancho del pilar p.

5.1.2. Variable de decisión y parámetros de entrada

La variable de decisión para la resolución del problema de optimización corresponde a:

$$x_i = \begin{cases} 1, & Si\ el\ caserón\ i\ se\ extrae \\ 0, & Caso\ contrario \end{cases}$$

Ecuación 4: Variable de decisión- Generación Envolvente Óptima

Por su parte, los parámetros de entrada requeridos para el modelo, se presentan a continuación:

B_i: Beneficio asociado a la extracción del caserón i

5.1.3. Función objetivo

La función objetivo para el problema de generación de la envolvente óptima de caserones vendrá dada por la maximización del beneficio de la extracción:

Maximizar el BENEFICIO

$$\sum_{i \in N} x_i B_i$$

Ecuación 5: Función objetivo- Generación Envolvente Óptima

5.1.4. Restricciones

Las ecuaciones que permiten modelar el problema de optimización expuesto, corresponden a las restricciones que le entregan sentido minero geométrico y geotécnico a la explotación por medio de minería selectiva auto-soportada. Éstas se presentan a continuación:

1. Restricción que prohíbe traslape entre caserones:

$$x_i + x_{i'} \leq 1, \quad \forall\, i \in N, \quad \forall\, i' \in bc_i$$

Ecuación 6: Restricción producción caserones traslapados- Modelo Envolvente Económica

2. Restricción que prohíbe la extracción de caserones que no comparten niveles de extracción en común (piso de extracción en común):

$$x_i + x_{i'} \leq 1, \quad \forall\, i \in N, \quad \forall\, i' \in cn_i$$

Ecuación 7: Restricción producción caserones distinto nivel- Modelo Envolvente Económica

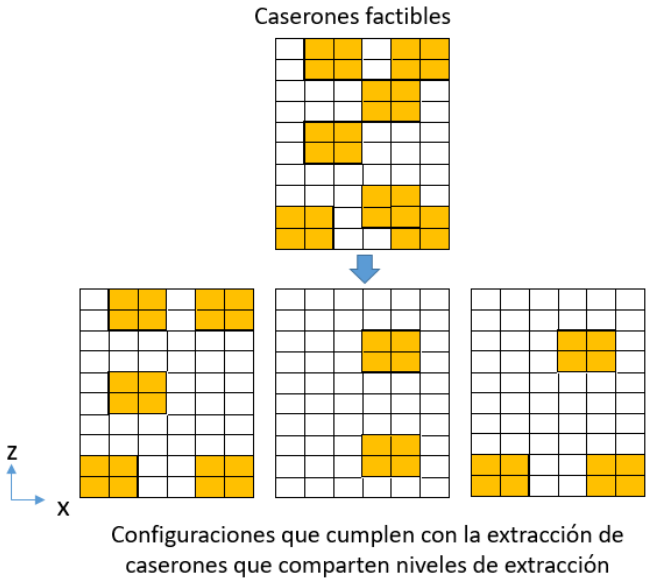


Ilustración 8: Ejemplo configuraciones para la extracción de caserones en mismo nivel

Como es posible observar en la ilustración anterior, de un determinado modelo de caserones factibles, existirán diversas configuraciones que cumplen con seleccionar aquellos caserones que se encuentran en el mismo nivel de extracción. La selección de éstos irá acorde a la función objetivo (maximización del beneficio económico).

3. Restricción que prohíbe la extracción de caserones que no se encuentran ubicados a una distancia igual o superior al ancho del pilar:

$$x_i + x_{i'} \leq 1, \quad \forall\, i \in N, \quad \forall\, i' \in ca_i$$

Ecuación 8: Restricción producción caserones sin pilar- Modelo Envolvente Económica

4. Restricción para variables binarias:

$$x_i, x_{i'} \in [0,1] \quad \forall\, i, i'$$

Ecuación 9: Restricción variables binarias- Modelo Envolvente Económica

5.2. Modelo lineal para comparación con algoritmo Sandanayake

Como se presentó en la sección 4.1.2, el algoritmo de Sandanayake incorpora los niveles asociados a la losa (o Crown Pillar) mediante una búsqueda de los caserones factibles en la coordenada vertical asociados a un valor equivalente a la altura máxima de caserón. Dado lo anterior, para comparar los resultados obtenidos por ese método y el propuesto, así como la eficiencia computacional, es necesario entregarle al modelo de optimización lineal el mismo archivo de entrada. Es decir, el archivo de caserones con niveles predefinidos corresponde al input del problema.

Teniendo lo anterior en cuenta, la restricción asociada a la producción de caserones en niveles ubicados en distinta cota z (Ecuación 7) no es necesaria. De esta forma es posible tener soluciones que sean comparables con la heurística de la sección anterior.

Dado el modelo propuesto en la sección 5.1, es de esperar que los resultados que entregue éste sean asociados a un beneficio económico superior o igual a aquellos obtenidos con los niveles predefinidos. En la Ilustración 9 se presenta un esquema general de los modelos a evaluar según los parámetros de entrada respectivos.

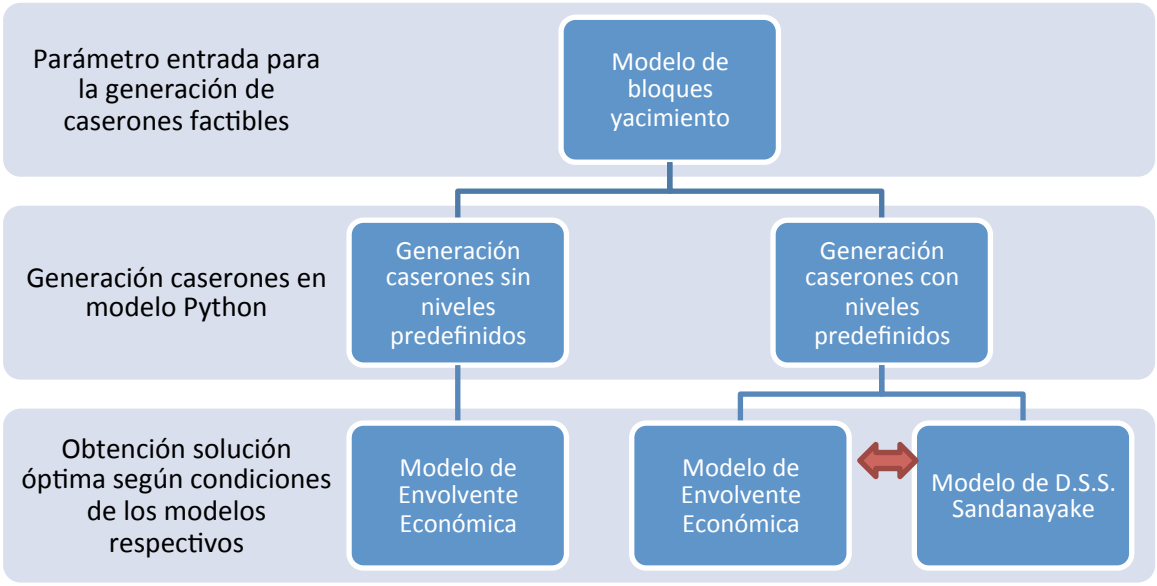


Ilustración 9: Esquema general modelos de optimización evaluados

5.3. Casos de estudio

Como casos de estudio para la validación del modelo de optimización, y obtención de resultados que permitan evaluar la eficiencia computacional, se determinó realizar 5 modelos que varían en la cantidad de caserones factibles que ingresan a la optimización. A su vez, éstos consisten en 10 realizaciones distintas asociadas a modelos de bloques sintéticos con variaciones en las leyes del mineral, y con tamaño de bloques de 10[m]x10[m]x10[m]. De esta forma, se tienen 50 corridas. Es de resaltar el hecho que los modelos principales varían en las dimensiones de yacimiento (distintas cantidades de bloques), y que se realizan para un tamaño de caserón de 30[m]x30[m]x30[m] con un ancho de pilar de 10[m]; con el fin principal de evaluar la eficiencia de los modelos. Con estos datos de entrada es posible generar los caserones posibles según lo explicado en la sección 4.1.1 y 4.1.2.

En la Tabla 2, se muestran los distintos casos de estudio a ingresar al algoritmo de Sandanayake y el modelo de optimización lineal para la obtención del layout de caserones con niveles predefinidos. Los parámetros económicos utilizados se presentan en la Tabla 13. Por otro lado, en la Tabla 15 se encuentran las características principales de los modelos de bloques utilizados, los cuales varían principalmente en las dimensiones del yacimiento (Tabla 14) y las leyes de cobre promedio. A modo de esquematizar y resumir el estudio en cuestión, se tiene la Ilustración 10.

De igual forma, los tiempos de generación de caserones para cada modelo de bloques y caso a estudiar, se presentan en la sección anexos 10.6. En estos se presentan los tiempos asociados a la lectura de los modelos de bloques y la correspondiente generación, según los parámetros de entrada entregados.

Tabla 2: Casos de estudio – Niveles predefinidos

		Caserones factibles de entrada al modelo de optimización con niveles predefinidos									
		M1	**M2**	**M3**	**M4**	**M5**	**M6**	**M7**	**M8**	**M9**	**M10**
Caso 1	**N° cas**	2,352	2,352	2,352	2,352	2,352	2,352	2,352	2,352	2,352	2,352
	Ley [%]	0.64	1.61	0.96	1.32	1.96	0.64	2.30	1.48	0.89	1.09
Caso 2	**N° cas**	3,920	3,920	3,920	3,920	3,920	3,920	3,920	3,920	3,920	3,883
	Ley [%]	1.47	2.08	0.97	1.60	1.53	0.96	1.44	2.41	1.78	0.63
Caso 3	**N° cas**	6,534	6,534	6,534	6,534	6,534	6,534	6,534	6,534	6,534	6,534
	Ley [%]	1.61	1.29	0.79	0.80	1.05	1.68	1.55	0.68	0.86	1.38
Caso 4	**N° cas**	8,300	8,329	7,993	8,194	8,138	8,174	8,286	8,150	8,258	8,247
	Ley [%]	1.08	0.77	0.77	0.92	0.59	2.04	0.62	0.79	2.54	1.64
Caso 5	**N° cas**	10,108	10,108	10,108	10,108	10,108	10,108	10,108	10,108	10,108	10,108
	Ley [%]	1.60	1.32	1.12	0.92	0.64	0.84	0.79	1.92	1.33	1.08

Es importante mencionar que el modelo lineal libre (sin considerar niveles predefinidos) debería entregar un resultado superior o igual a aquél encontrado en la optimización con niveles debido a que tiene mayor libertad a la hora de determinar la configuración de niveles y caserones a extraer. Para probar esto, se

tiene el caso de estudio de la Tabla 3 . Se busca comparar el resultado del modelo lineal libre con aquél obtenido por la heurística de Sandanayake y el modelo lineal con niveles.

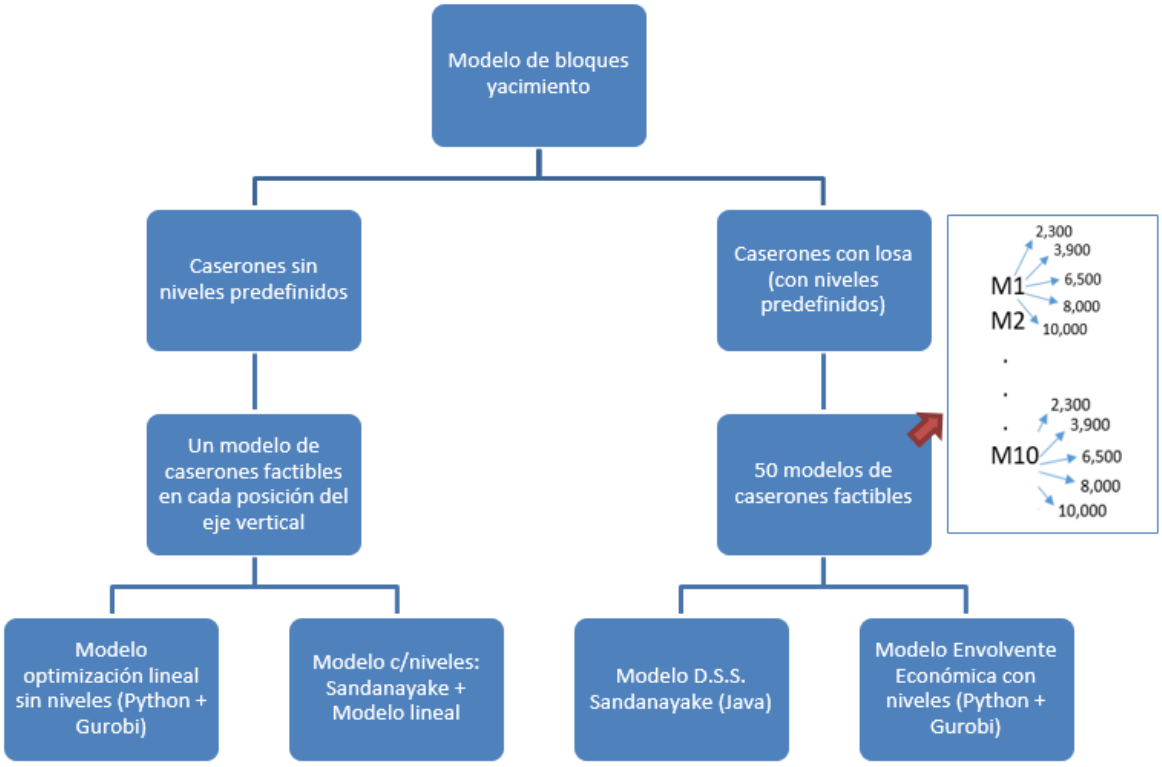


Ilustración 10: Esquema general Casos Estudio

Tabla 3: Caso Estudio General – Sin Niveles Predefinidos

	Caserones s/nivel	Caserones c/nivel
N° caserones posibles	4,096	768
Ley promedio caserones posibles [%]	0.64	0.64

Lo anterior fue obtenido de correr el archivo Python determinado en un computador Intel® Core™ i7-4720HQ CPU @2.60 GHz con 12 GB de memoria RAM. Por último, en relación a los tiempos de generación de caserones, se presentan a continuación las estadísticas básicas obtenidas para los caserones con niveles predefinidos. En el caso de estudio que involucra caserones sin predefinición de niveles, se muestran los tiempos incurridos en la Tabla 5.

Se puede observar, que el tiempo requerido para generar los caserones es bajo, en orden de segundos. De igual forma, como es de esperarse, es posible notar que el tiempo de generación incrementa en la medida que aumenta la cantidad de bloques del modelo, lo que se observa principalmente en el aumento de tiempo del Modelo 1 al Modelo 5 (con niveles predefinidos). En relación éste, se tiene una diferencia de 17.41 [seg] en promedio entre el caso 1 (menor cantidad de bloques) y caso 5 (mayor cantidad de bloques).

Tabla 4: Estadísticas básicas tiempos generación caserones – Niveles predefinidos

		Promedio [seg]	Desv. Std [seg]	Mín [seg]	Máx [seg]
Modelo 1 (c/niveles)	**Lectura mdb**	3.23	0.13	2.90	3.36
	Generación caserones	0.40	0.04	0.31	0.43
	Total	**3.63**	**0.16**	**3.22**	**3.77**
Modelo 2 (c/niveles)	**Lectura mdb**	5.08	0.66	3.66	5.83
	Generación caserones	0.64	0.12	0.41	0.82
	Total	**5.72**	**0.76**	**4.07**	**6.47**
Modelo 3 (c/niveles)	**Lectura mdb**	8.66	0.25	8.32	9.08
	Generación caserones	1.21	0.05	1.12	1.30
	Total	**9.87**	**0.28**	**9.54**	**10.34**
Modelo 4 (c/niveles)	**Lectura mdb**	13.84	1.03	11.96	15.83
	Generación caserones	1.74	0.16	1.38	1.93
	Total	**15.58**	**1.14**	**13.33**	**17.60**
Modelo 5 (c/niveles)	**Lectura mdb**	13.49	1.11	11.89	14.95
	Generación caserones	1.73	0.17	1.38	1.90
	Total	**15.22**	**1.18**	**13.33**	**16.49**

Tabla 5: Tiempo generación caserones factibles – Caso Estudio General

Tiempo generación de caserones factibles [seg]					
Sin nivel			**Con nivel**		
Lectura	**Generación**	**Total**	**Lectura**	**Generación**	**Total**
1.22	1.02	**2.23**	1.17	0.19	**1.36**

En cuanto a los tiempos de generación de caserones para el Caso de Estudio General, dado que la cantidad de caserones input es de aproximadamente 4,000, el tiempo incurrido es bajo para la generación sin y con niveles.

6. RESULTADOS OPTIMIZACIÓN

A partir de lo presentado en secciones anteriores, se presentan a continuación los resultados obtenidos del estudio. Las características del computador donde fueron corridos los modelos de optimización se presentan en la Tabla 6.

Tabla 6: Características computadores – Corridas Casos Estudio

Modelo	Procesador	RAM
D.S.S. Sandanayake + Lineal libre	Intel® Core™ i7-2600 CPU @3.40GHz, 8 procesadores (Ubuntu)	12 GB
Lineal con niveles + Sandanayake Caso General	Intel® Core™ i7-4720HQ CPU @2.60 GHz, 8 procesadores (Windows)	12 GB

6.1. Algoritmo Sandanayake VS. Modelo lineal con niveles

6.1.1. Resultados optimización

Con el fin de estudiar la heurística propuesta por D.S.S. Sandanayake (2014) se realizaron las corridas explicadas en la sección 5.3 para tres valores de Ctk: 10,100 y 1000. Como se explicó en la sección 4.1.3, Ctk corresponde a la cota impuesta para el total de familias de sets de caserones únicos que cumplen con no traslapar entre sí, y también se encuentran a una distancia mínima de 10 [m] (correspondiente al pilar). A partir de ello, se tienen los resultados que se presentan de forma siguiente.

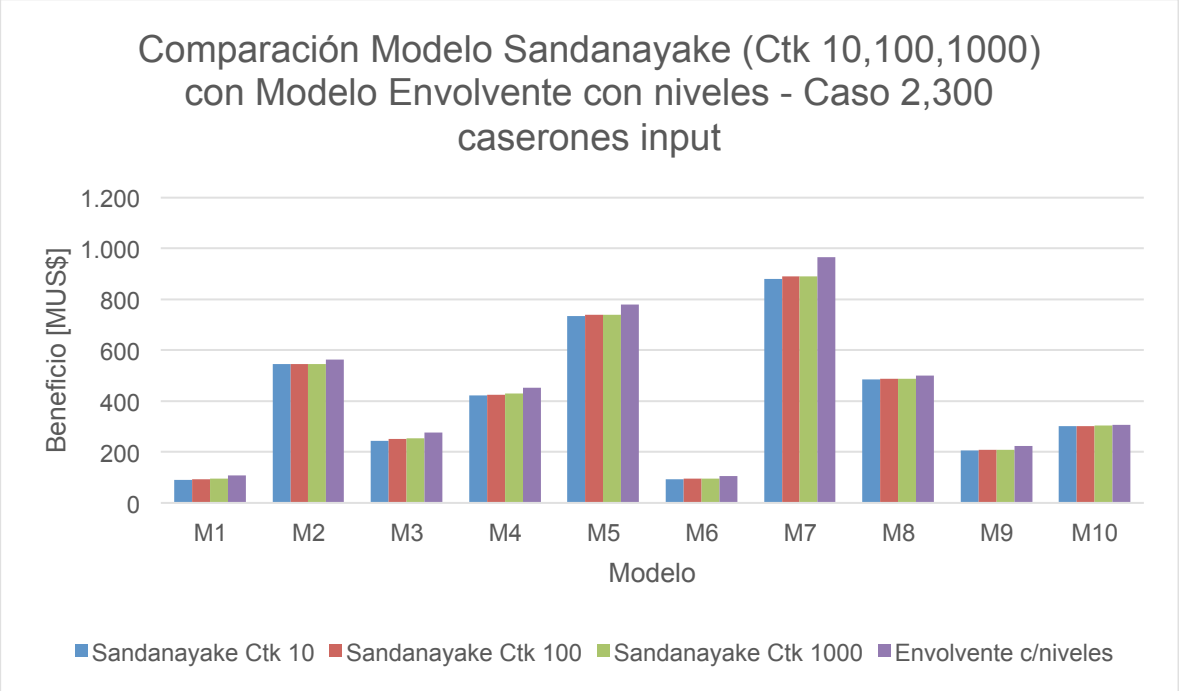


Ilustración 11: Comparación beneficio Modelo Sandanayake e IP con niveles – 2,300 cas

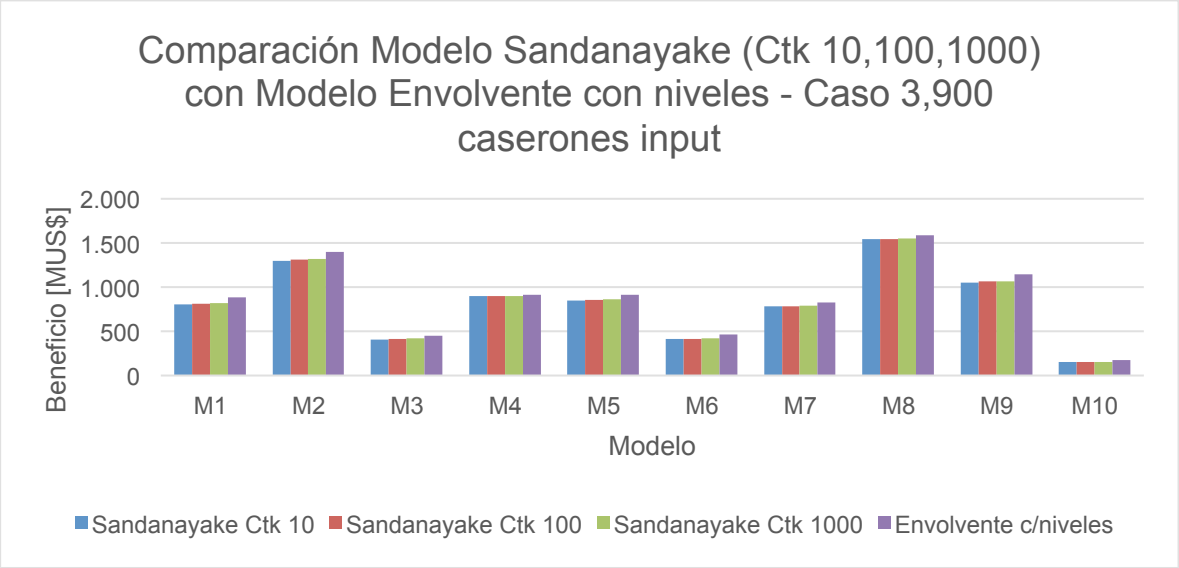


Ilustración 12: Comparación beneficio Modelo Sandanayake e IP con niveles – 3,900 cas

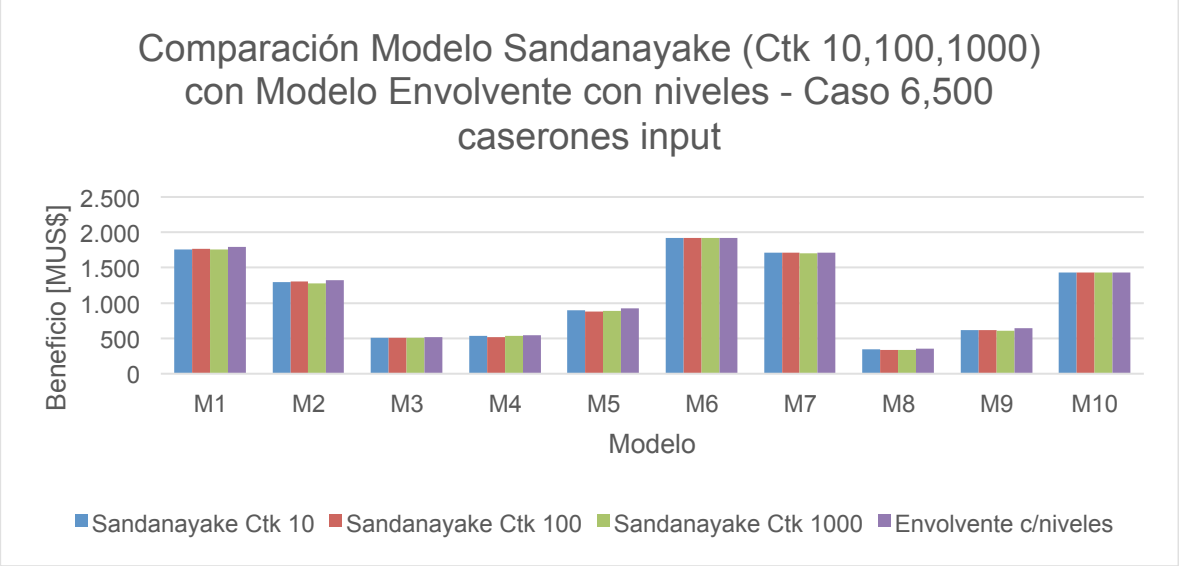


Ilustración 13: Comparación beneficio Modelo Sandanayake e IP con niveles – 6,500 cas

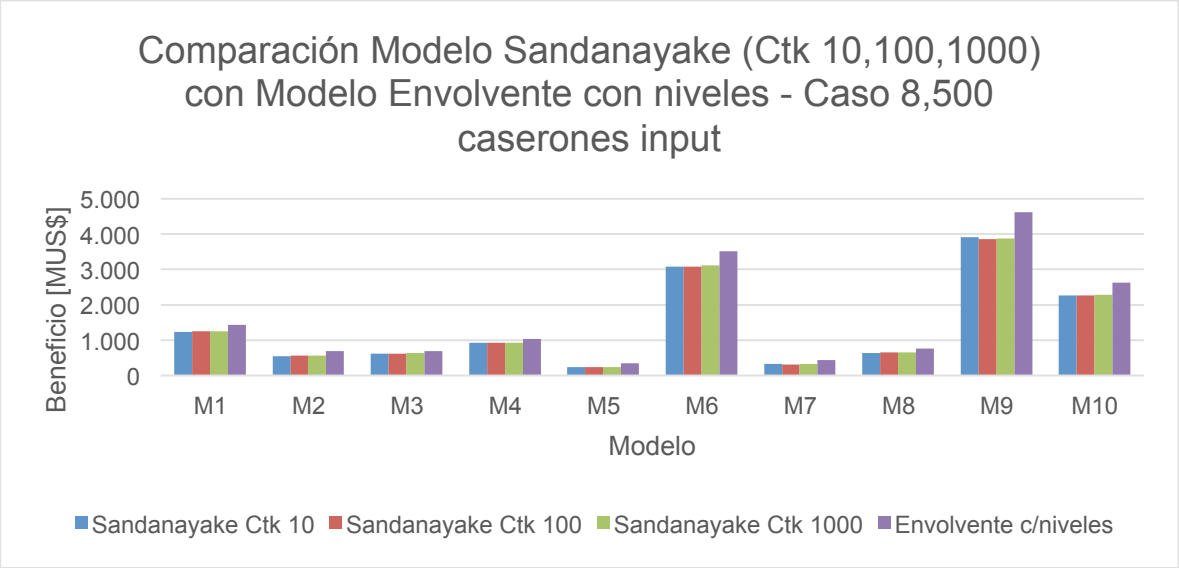


Ilustración 14: Comparación beneficio Modelo Sandanayake e IP con niveles – 8,500 cas

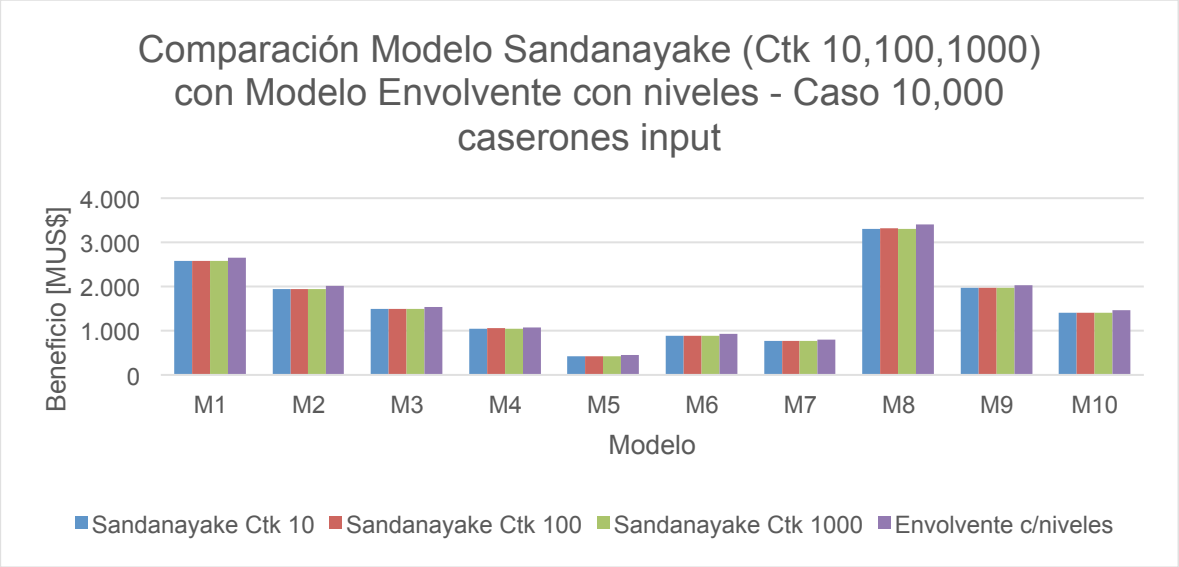


Ilustración 15: Comparación beneficio Modelo Sandanayake e IP con niveles – 10,000 cas

Primero que todo, en cada caso se observa la clara variación en beneficio económico entre las corridas realizadas (M1-M10). Esto, es principalmente debido al efecto de la ley media de los modelos. Se puede notar; como es de esperarse; que a mayor ley del modelo de bloques (por ende, del modelo de caserones) el beneficio asociado es mayor para el mismo (o similar) número de caserones que ingresan a la optimización.

Por otro lado, de los gráficos anteriores es posible comparar los resultados obtenidos según el algoritmo de Sandanayake – para los tres Ctk impuestos – junto con el modelo lineal con niveles predefinidos. Se puede observar que en todos los casos el resultado del modelo de optimización IP entrega mejores o

iguales resultados, de hecho, esto se cumple en el 96% de los casos. Esto se puede
observar en la Ilustración 17 a Ilustración 21 donde se grafica la variación porcentual (como se presenta en la Ecuación 10) del modelo heurístico con respecto al modelo lineal con niveles, en función del parámetro heurístico -Ctk- impuesto al algoritmo.

$$Gap\ [\%] = \frac{Valor_{óptimo}[MUS\$] - Valor_{estimado}[MUS\$]}{Valor_{óptimo}[MUS\$]}$$

Ecuación 10: Estimación gap

De igual forma, en la Tabla 7 se presentan las estadísticas básicas asociadas, y en la Ilustración 16 se grafica la variación porcentual promedio (gap) para cada Ctk y caso evaluado, en conjunto con una barra de error que indica el worst case (valores máximos) y best case (valores mínimos) obtenidos. Asimismo, en el eje secundario se presentan las iteraciones promedio realizadas por el algoritmo de Sandanayake para la obtención del resultado óptimo. En la sección anexos 10.8 se presentan las iteraciones para cada modelo y caso evaluado.

Tabla 7: Estadísticas básicas gap por Ctk

		Gap Beneficio Sandanayake vs Modelo lineal c/niveles				
		Caso 1	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**
Ctk 10	**Promedio**	7.80%	7.73%	1.92%	16.63%	3.83%
	Desv. Std.	4.98%	4.04%	1.58%	6.23%	1.70%
	Mínimo	1.53%	1.64%	0.00%	9.80%	2.42%
	Máximo	17.43%	15.73%	4.49%	28.53%	8.17%
Ctk 100	**Promedio**	6.70%	7.21%	2.48%	16.53%	3.70%
	Desv. Std.	4.07%	3.85%	2.33%	6.11%	1.42%
	Mínimo	1.53%	1.64%	0.00%	10.53%	2.42%
	Máximo	14.68%	15.17%	6.03%	27.65%	6.84%
Ctk 1000	**Promedio**	6.40%	6.63%	2.38%	15.69%	3.64%
	Desv. Std.	3.71%	34.05%	2.19%	5.86%	1.42%
	Mínimo	1.53%	1.53%	0.00%	9.37%	2.35%
	Máximo	13.76%	14.61%	6.15%	27.65%	7.06%

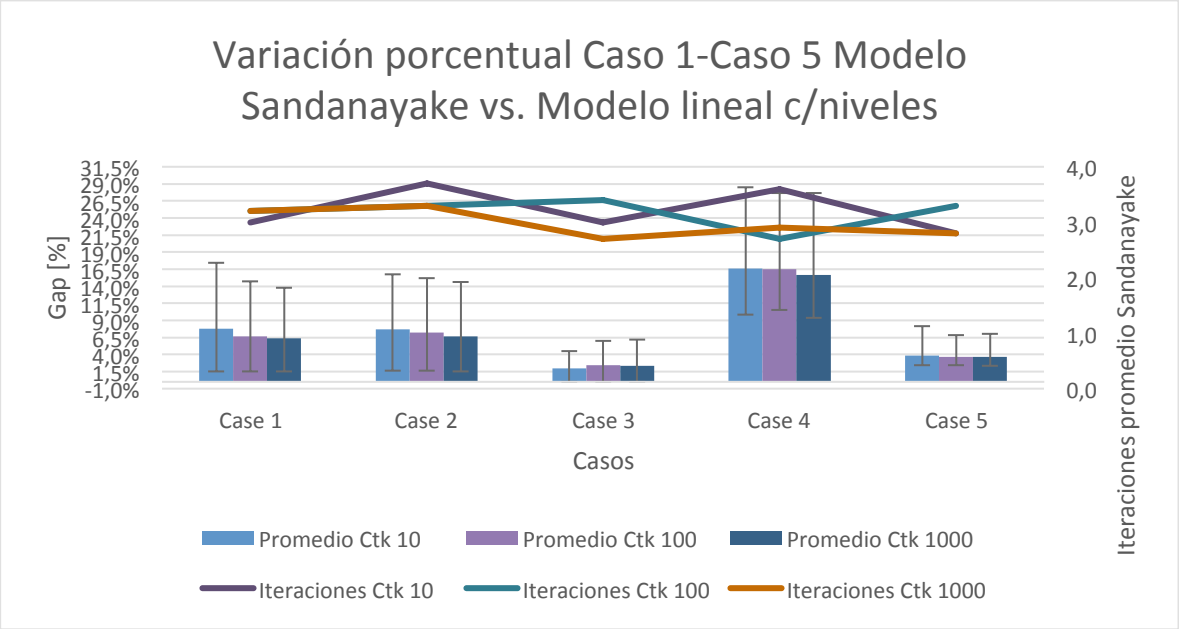


Ilustración 16: Gap promedio solución Sandanayake

En términos de las iteraciones incurridas por el algoritmo heurístico, se observa que éstas en promedio son 3 (para cada Ctk evaluado), y mínimo 2 iteraciones (como es de esperar por el funcionamiento del algoritmo). La máxima cantidad de iteraciones, por su parte, es función del Ctk. Se observa que para el menor valor (Ctk 10), en promedio se realizan 6 iteraciones sobre todos los caserones. A este número le sigue el Ctk intermedio, con un promedio de 5.8 iteraciones. Por último, el Ctk mayor presenta un promedio de 4.6 iteraciones. Lo anterior es de esperarse teniendo en consideración que, para una menor cota en la cantidad de sets únicos de caserones, el campo de búsqueda de la mejor solución es menor; por ende, mayor cantidad de iteraciones es requerida. De forma inversa se tiene para el Ctk 1000, en el cual para una iteración se presenta una mayor cantidad de sets factibles.

Por otro lado, es posible observar una tendencia a la disminución gap en la medida que aumenta el Ctk utilizado, en el caso anterior. Asimismo, es posible notar que entre los modelos heurísticos con distintos Ctk la diferencia porcentual no es mucha. Lo anterior se puede observar en las tablas de la sección 10.7, en donde se presentan estadísticas básicas del gap para cada modelo y caso estudiado, asociadas a la variación de Ctk. Específicamente, la baja desviación estándar indica que entre valor de Ctk estudiado (10,100 y 1000) no existen diferencias mayores, siendo la máxima diferencia de aproximadamente un 2.18% correspondiente al Caso 3 (6,500 caserones) modelo M4.

Es de resaltar el hecho que los valores anteriores se obtuvieron con el valor óptimo del modelo lineal obtenido del código Python y mediante el solver Gurobi. En relación al gap reportado por Gurobi para la solución óptima, en algunos casos no se obtuvo el resultado con un 0% de gap, esto ocurrió un 44% de las veces. Sin embargo, este valor siempre estuvo por debajo del 1%, siendo de 0.0030% en

promedio para el caso 1, 0.0116% para el caso 2, 0.00122% en el caso 3, y 0.0012% y 0.0022% para el caso 4 y 5; respectivamente. Lo anterior, en términos de la magnitud de beneficio económico asociado a las envolventes obtenidas, no es mayor considerando que dependiendo del caso se trata de un beneficio que está entre los 428 [MUS$] y 1,636 [MUS$] para el caso con menor y mayor cantidad de caserones, respectivamente.

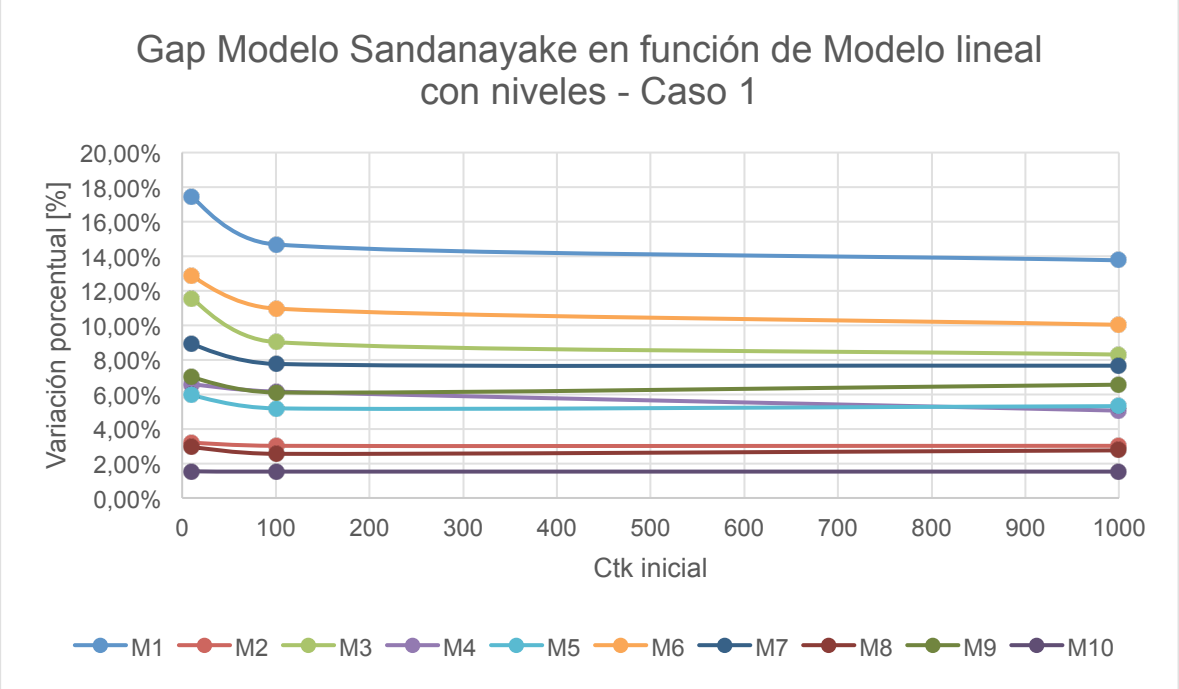


Ilustración 17: Gap porcentual Modelo Sandanayake en función Modelo IP – Caso 1

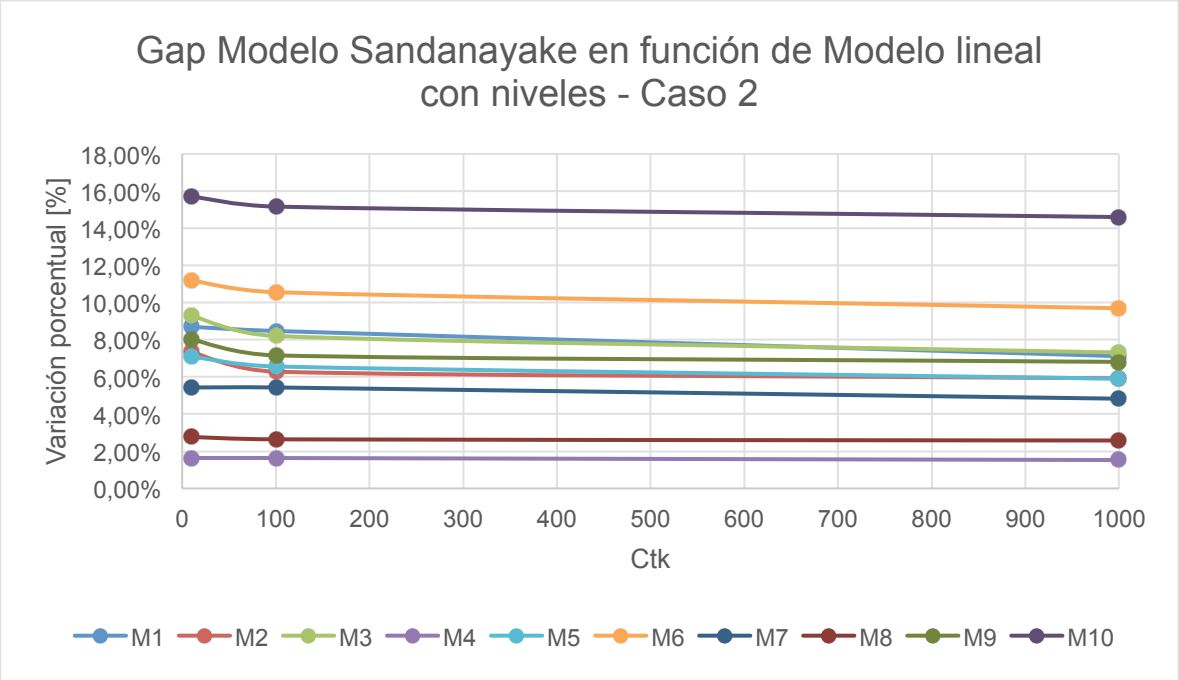


Ilustración 18: Gap Modelo Sandanayake en función Modelo IP – Caso 2

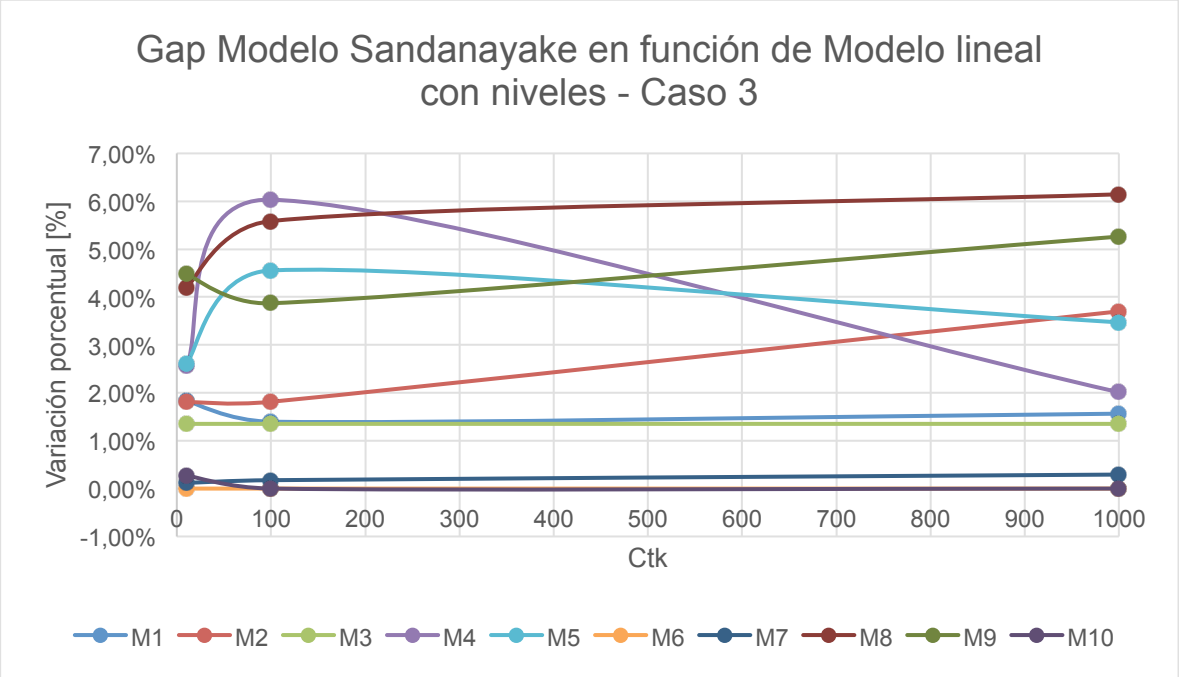


Ilustración 19: Gap Modelo Sandanayake en función Modelo Envolvente – Caso 3

Es posible notar en la Ilustración 19 que algunos modelos para el Ctk=100, presentan un incremento en la variación porcentual en relación al Ctk=10 (M4, M5 y M8). Esto puede deberse a cómo funciona el algoritmo de Sandanayake (Ilustración 51). Para un Ctk menor, en cada iteración de caserones, se preferirán las mejores familias de sets de caserones que no traslapan, asociado a un menor

tiempo de ejecución. Por otro lado, en la medida que aumenta el Ctk, al existir una mayor cantidad de familias únicas de caserones, el tiempo de ejecución aumenta, y es de esperar que el resultado mejore. Sin embargo, dado que la generación de los sets de caserones es iterativa y existen infinitas soluciones, se da lugar a que existan variaciones en los resultados obtenidos. Esto, ya que puede que la solución óptima que se espera no se encuentre para ciertos Ctk y, por tanto, se llegue a una especie de óptimo local. Lo anterior afirma el hecho que con el algoritmo de Sandanayake no es posible asegurar alcanzar el óptimo.

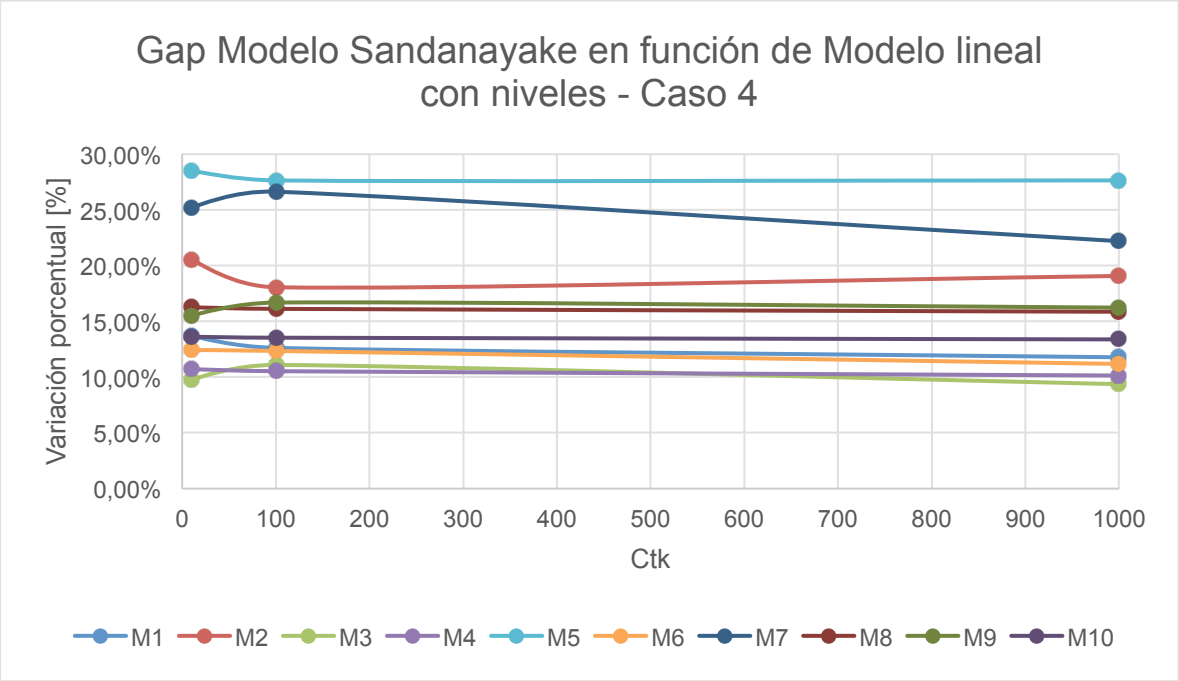


Ilustración 20: Gap Modelo Sandanayake en función Modelo Envolvente – Caso 4

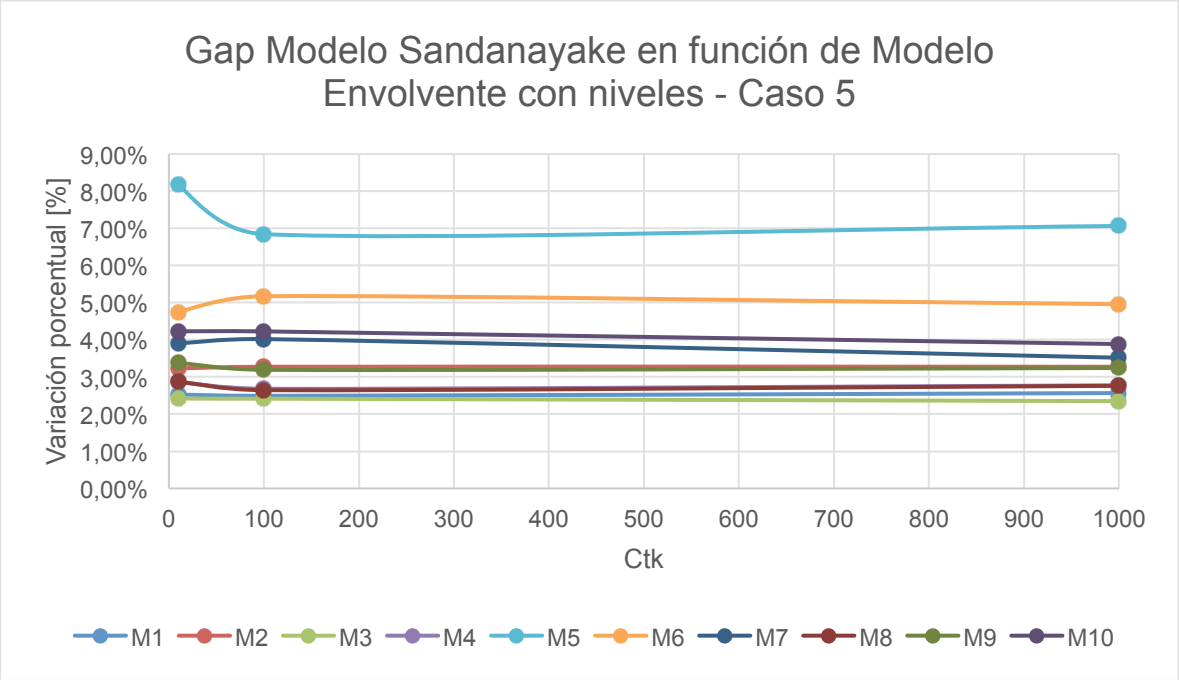


Ilustración 21: Gap Modelo Sandanayake en función Modelo Envolvente – Caso 5

Es de reiterar el hecho que estos modelos de optimización buscan la maximización del beneficio económico asociado a la extracción de los caserones que no se traslapan entre sí, cumplen con los niveles de extracción, así como con la distancia requerida por el ancho del pilar, el cual en todos los casos es de 10 [m].

A continuación, se presentan los gráficos asociados al resultado óptimo dado por el modelo lineal con niveles predefinidos para el Caso 1 – M6. Éste presenta 147 caserones en el layout óptimo, con una ley promedio de 0.68%, caserones de 30[m]x30[m]x30[m], y un beneficio total de extracción de 106.73 [MUS$]. Como es posible observar en la Ilustración 23 y Ilustración 24, no existe traslape de caserones en las tres cotas predefinidas (Cota 0, Cota 60 y Cota 120 [m]). Asimismo, los caserones se encuentran espaciados con mínimo el ancho del pilar impuesto (10 [m]).

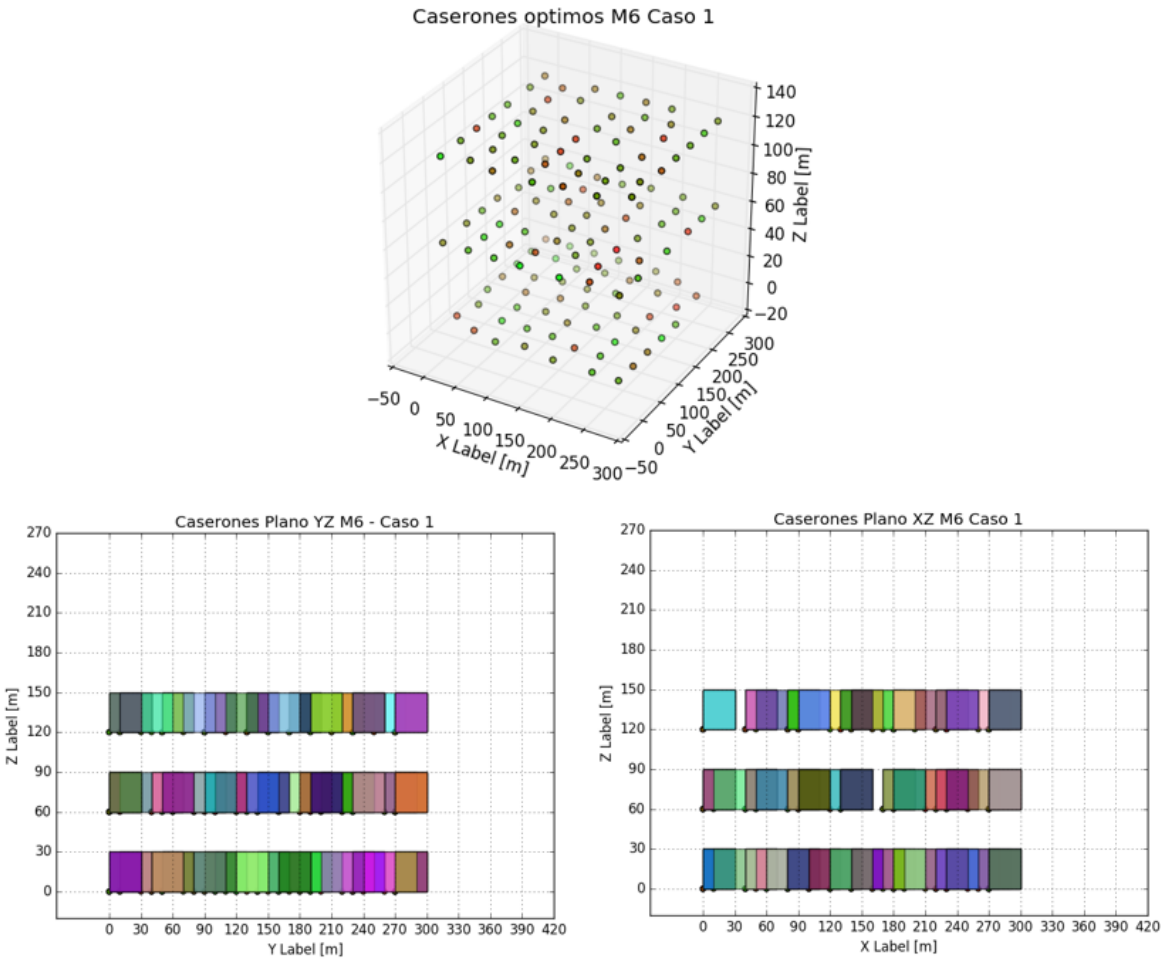


Ilustración 22: Caserones M6 Caso 1 Modelo lineal– 3D, Plano YZ, Plano XZ

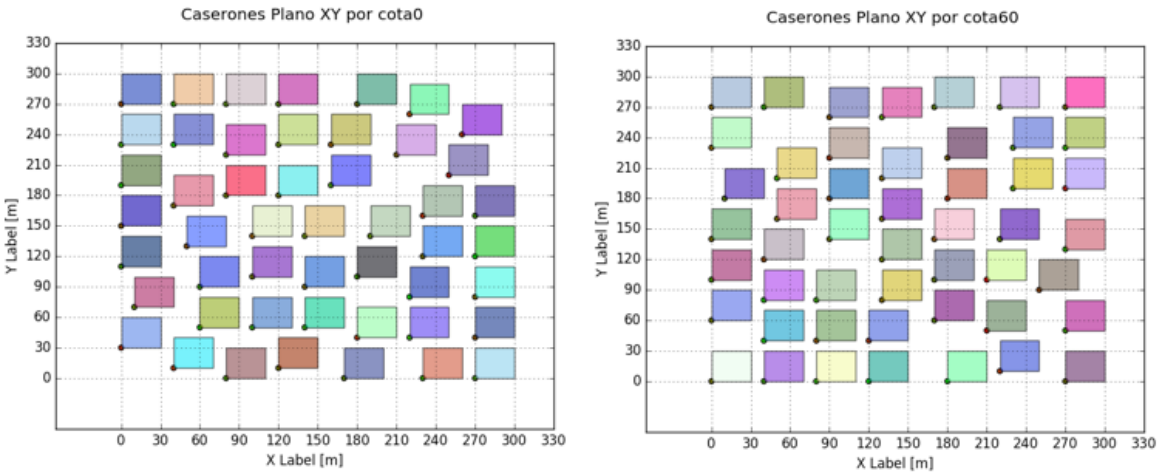


Ilustración 23: Caserones M6 Caso 1 Modelo lineal– Plano XY: Cota 0, Cota 60

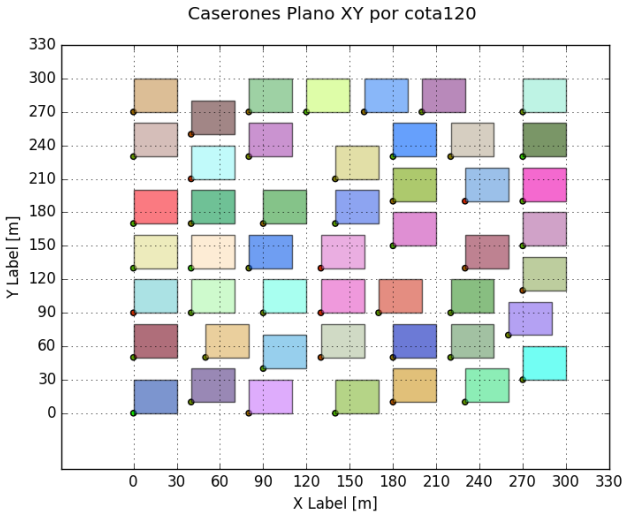
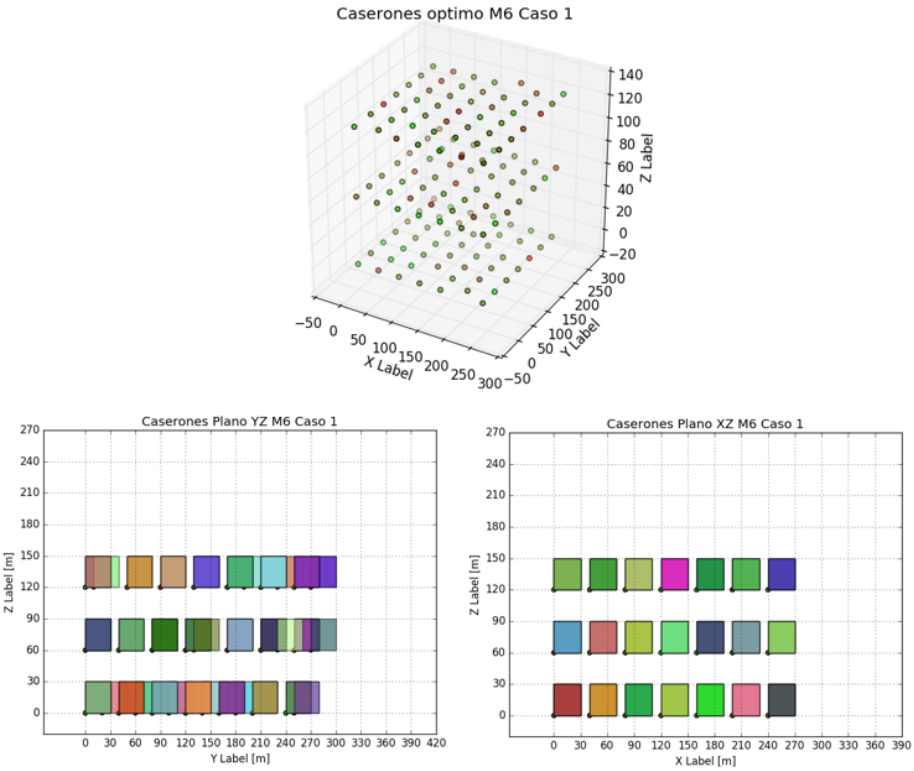


Ilustración 24: Caserones M6 Caso 1 Modelo lineal– Plano XY: Cota 120

De forma similar se tiene para el algoritmo de Sandanayake para cada Ctk evaluado.



.

Ilustración 25: Caserones M6 Caso 1 Sandanayake Ctk=10 – 3D, Plano YZ, Plano XZ

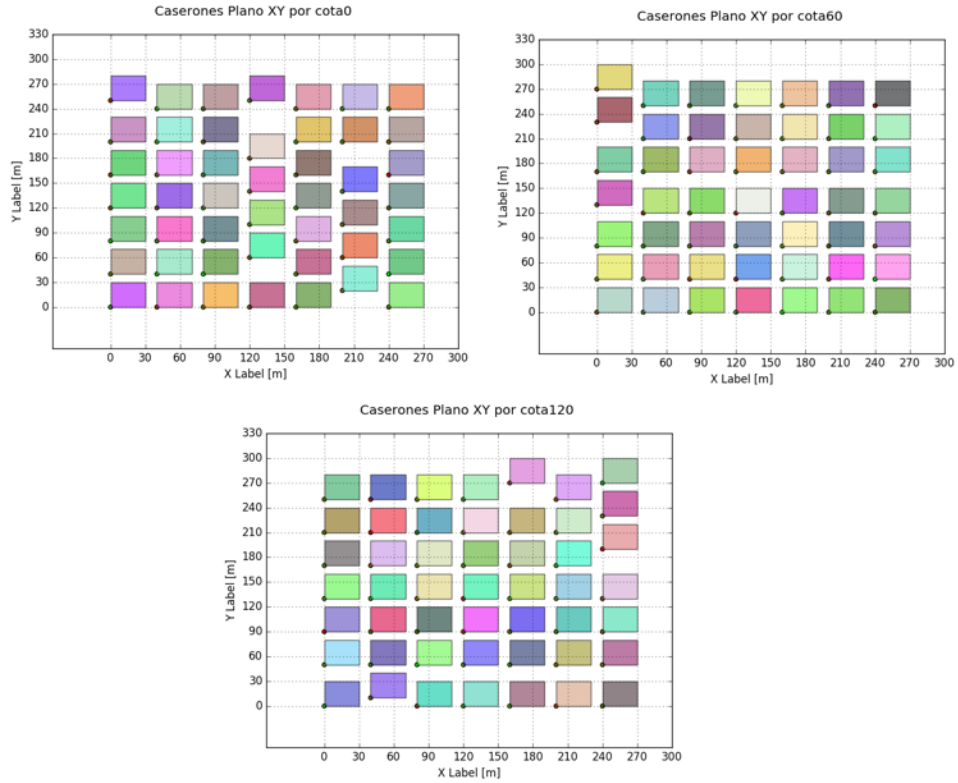


Ilustración 26: Caserones M6 Caso 1 Sandanayake Ctk=10 – Plano XY: Cota 0, Cota 60, Cota 120

Para el caso de Ctk=10, el resultado obtenido corresponde a un total de 145 caserones en el layout óptimo, con ley media de 0.65% y un beneficio de extracción de 92.58 [MUS$]. Por su parte, para el caso de Ctk=100, se tienen 146 caserones con una ley media de 0.66% y un beneficio económico de 95 [MUS$].

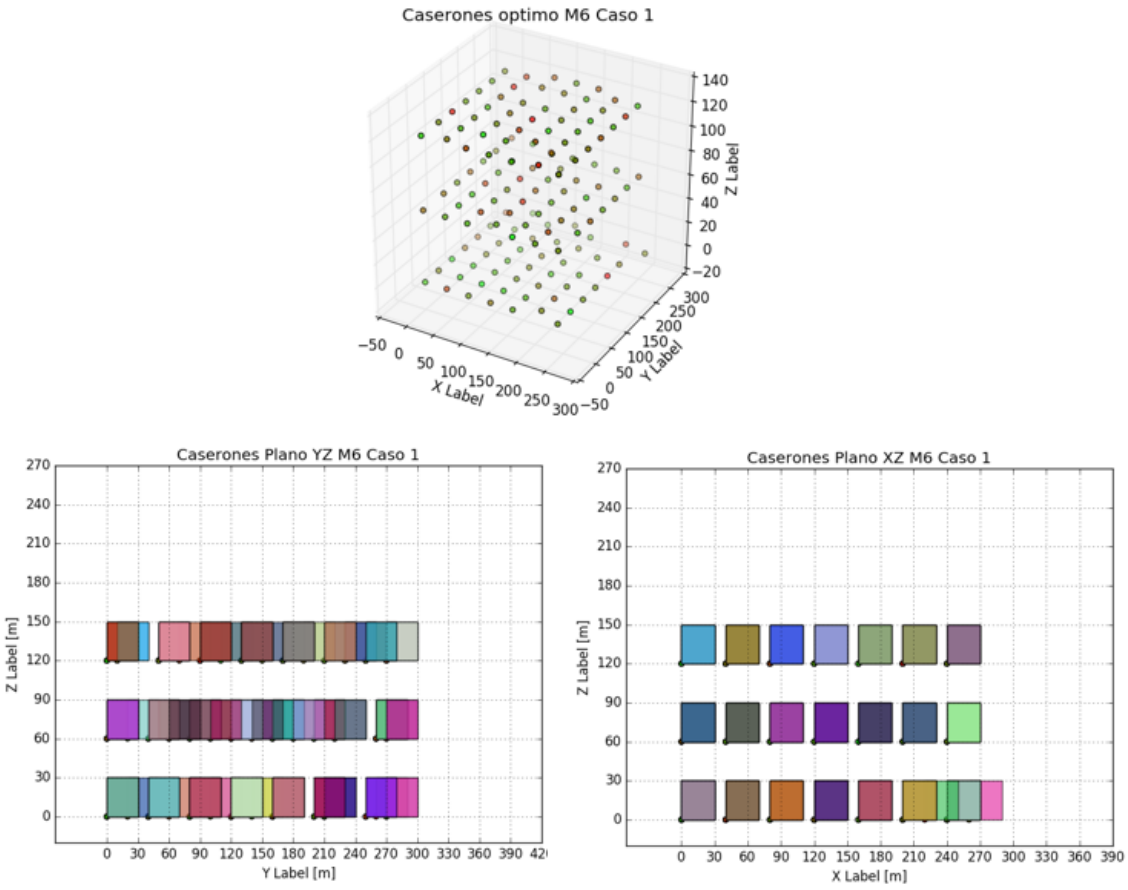


Ilustración 27: Caserones M6 Caso 1 Sandanayake Ctk=100 – 3D, Plano YZ, Plano XZ

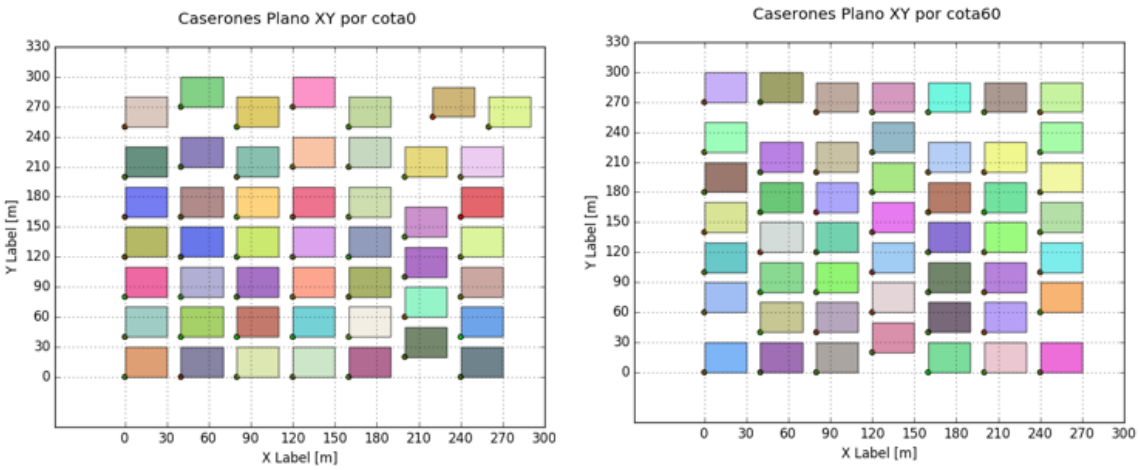


Ilustración 28: Caserones M6 Caso 1 Sandanayake Ctk=100 – Plano XY: Cota 0, Cota 60

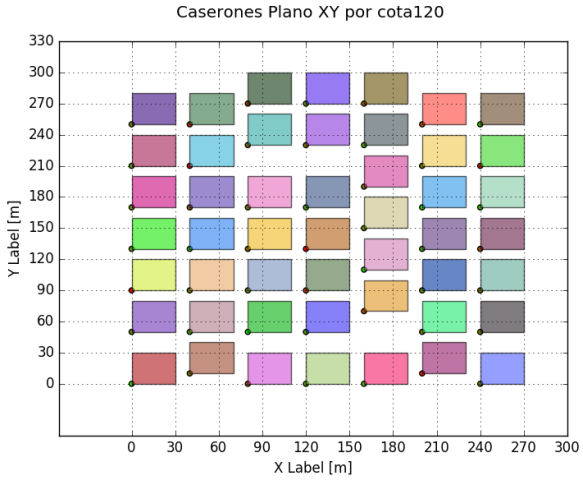


Ilustración 29: Caserones M6 Caso 1 Sandanayake Ctk=100 – Plano XY: Cota 120

Por último, en el caso de mayor Ctk (1000), se tiene un total de 147 caserones, asociado a una ley promedio de 0.66%, y un beneficio económico de extracción de 96.25 [MUS$]. Es posible notar que éstos difieren del óptimo logrado por el modelo lineal con niveles predefinidos, lo que es posible observar en los distintos layouts obtenidos en cada caso.

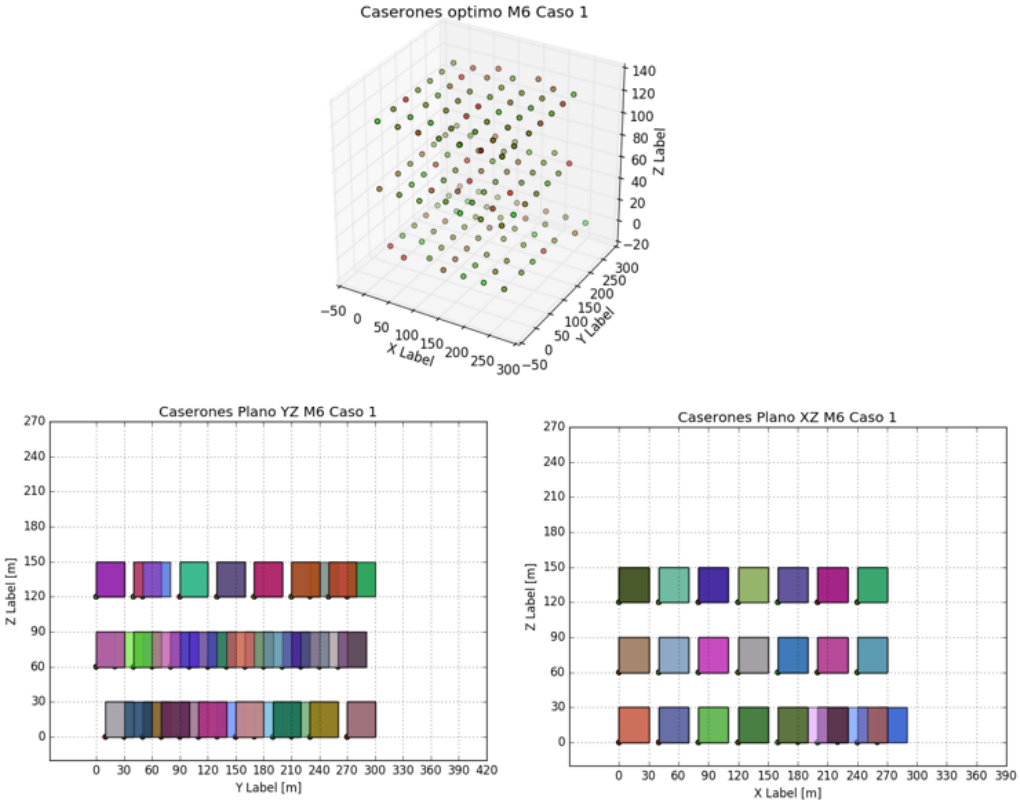


Ilustración 30: Caserones M6 Caso 1 Sandanayake Ctk=1000 – 3D, Plano YZ, Plano XZ

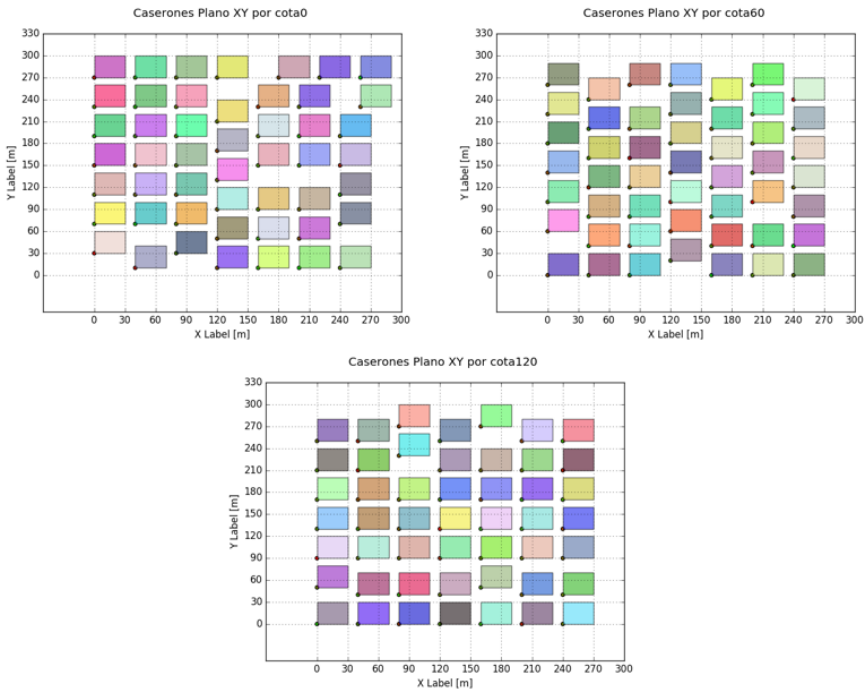


Ilustración 31: Caserones M6 Caso 1 Sandanayake Ctk=1000 – Plano XY: Cota 0, Cota 60, Cota 120

6.1.2. Tiempos optimización

En relación a los tiempos incurridos en la optimización, se presentan las gráficas a continuación, las que presentan los tiempos de ejecución asociados a la heurística de Sandanayake para tres valores de Ctk (10,100 y 1000). Éstas presentan el eje del tiempo en escala logarítmica con el fin de mostrar, de mejor forma, el comportamiento de éste en función de la cantidad de sets de caserones impuestos en la optimización. Es de resaltar el hecho que las heurísticas (Sandanayake) se corrieron en un computador de escritorio con las características de la Tabla 6, y por medio de un código programado en Java. Por otro lado, las corridas asociadas al modelo lineal para la envolvente óptima se realizaron en un computador portátil en paralelo a las heurísticas, y en código programado en Python.

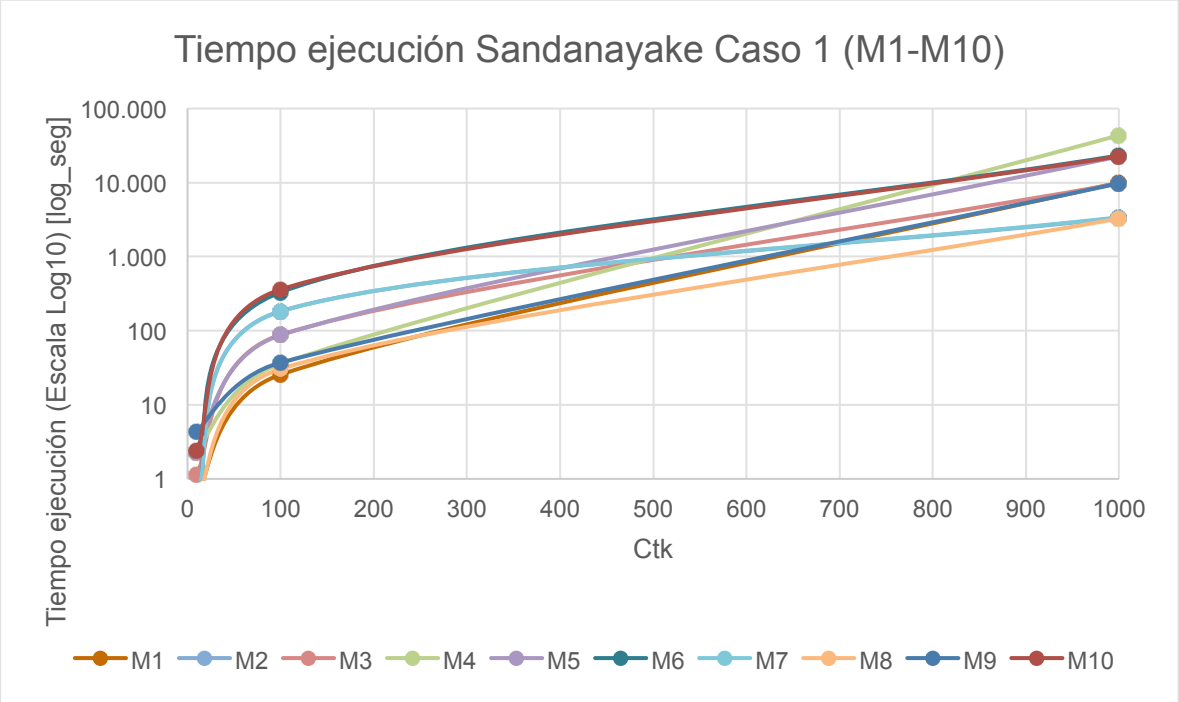


Ilustración 32: Tiempo ejecución modelo Sandanayake en función de Ctk – Caso 1

Se puede observar que para el Caso 1, el que posee menor cantidad de caserones input, el tiempo promedio de ejecución varía según el Ctk con un valor de 1.53 [seg] para Ctk=10, 136.35 [seg] para Ctk=100, y 15,175.30 [seg] para Ctk=1,000, valores que es posible extraer de la gráfica o de las tablas en la sección anexos 10.8. En otras palabras, en el caso que se permite una mayor cantidad de familias de caserones, la optimización llega a un tiempo promedio aproximado de 4.20 horas, siendo el caso de mayor tiempo obtenido en aproximadamente 12 horas.

Por su parte, en el Caso 2 se tiene que los tiempos asociados a la resolución se encuentran en promedio en 5.79 [seg] para el menor Ctk, y 40,932.28 [seg] (11.37 horas) para el mayor Ctk. El mayor tiempo recolectado corresponde a 28.26 horas,

lo que aumentó de forma considerable en relación al primer caso. Esto, considerando un aumento cercano a los 1,500 caserones.

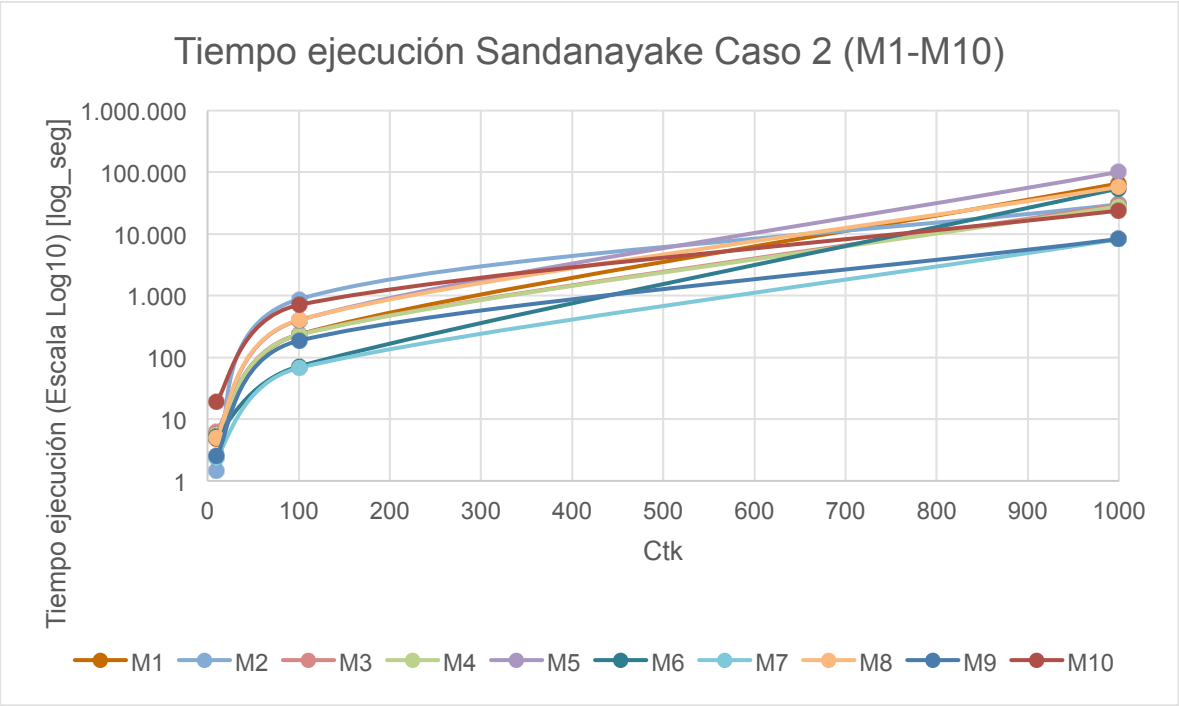


Ilustración 33: Tiempo ejecución modelo Sandanayake en función de Ctk – Caso 2

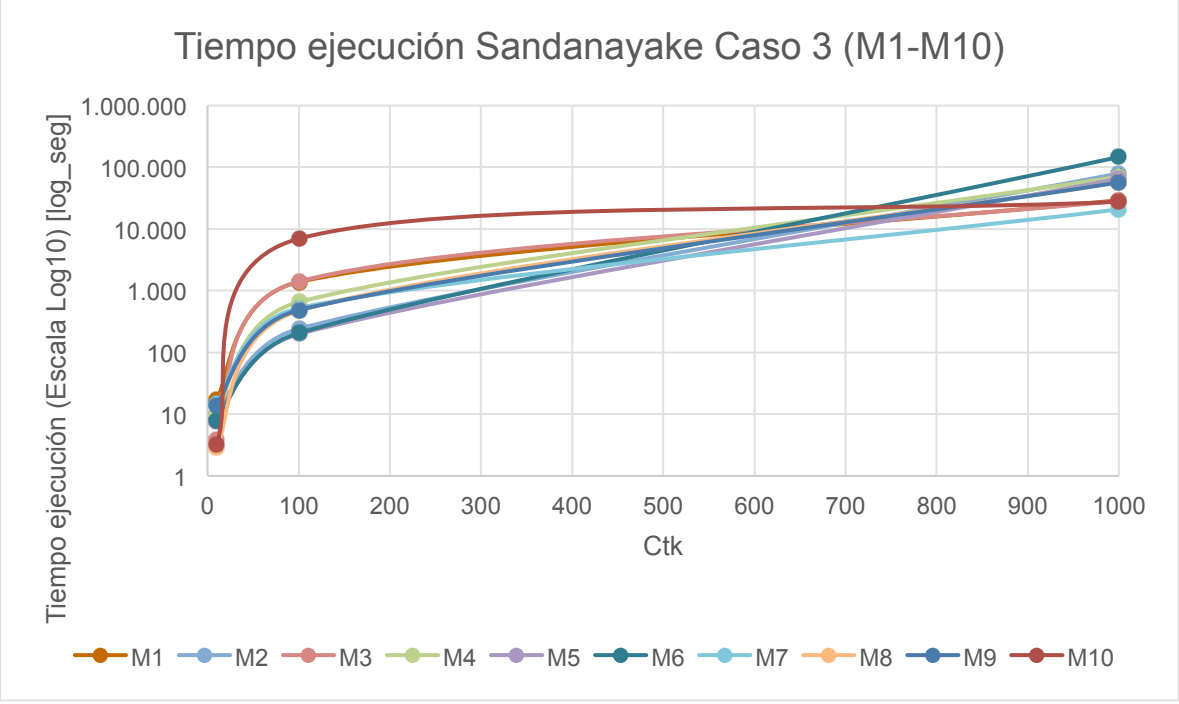


Ilustración 34: Tiempo ejecución modelo Sandanayake en función de Ctk – Caso 3

De forma similar se tiene para el Caso 3 donde el tiempo de ejecución promedio para Ctk=10 es bajo (en promedio 9.01 [seg]); sin embargo, aumenta de forma considerable para el Ctk=100, y en mayor medida para el caso Ctk=1000. Esto se respalda con valores promedio de 1,252.92 [seg] y 58,236.98 [seg] (16.18 hrs]), siendo el último número correspondiente al Ctk mayor. En relación al tiempo de ejecución máximo se tiene que para el Ctk de menor valor son aproximadamente 17 [seg]; sin embargo, con Ctk=1000, el valor asciende a las 40.8 horas.

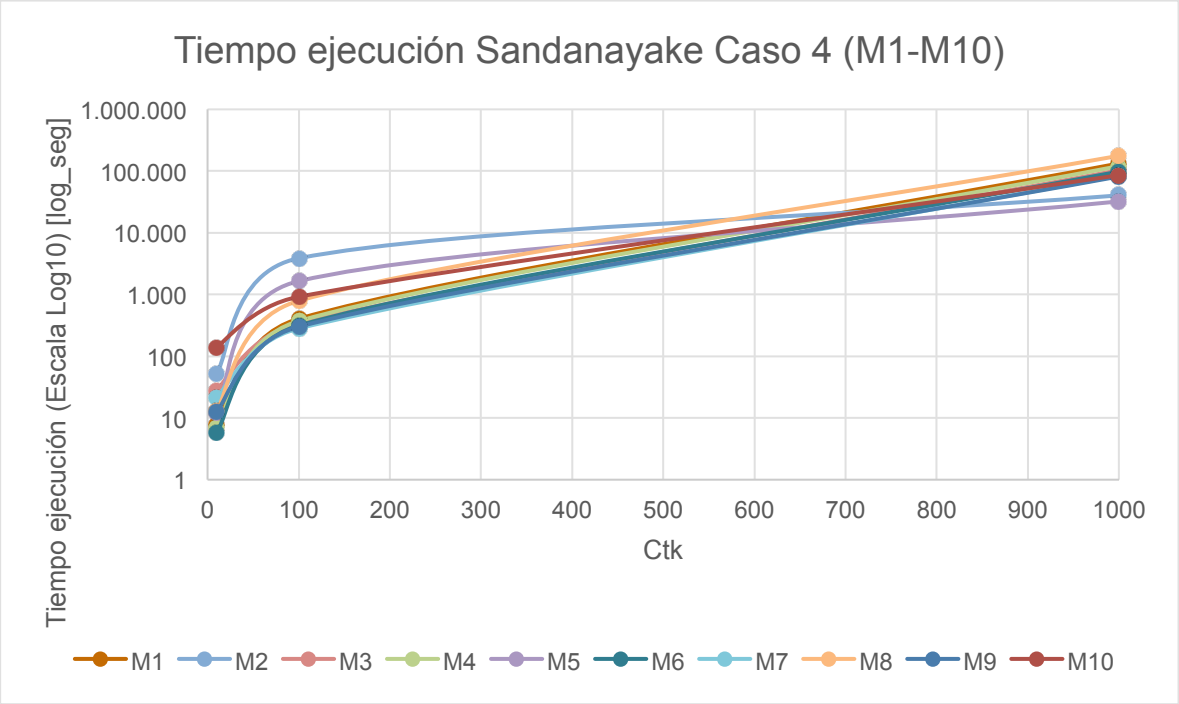


Ilustración 35: Tiempo ejecución modelo Sandanayake en función de Ctk – Caso 4

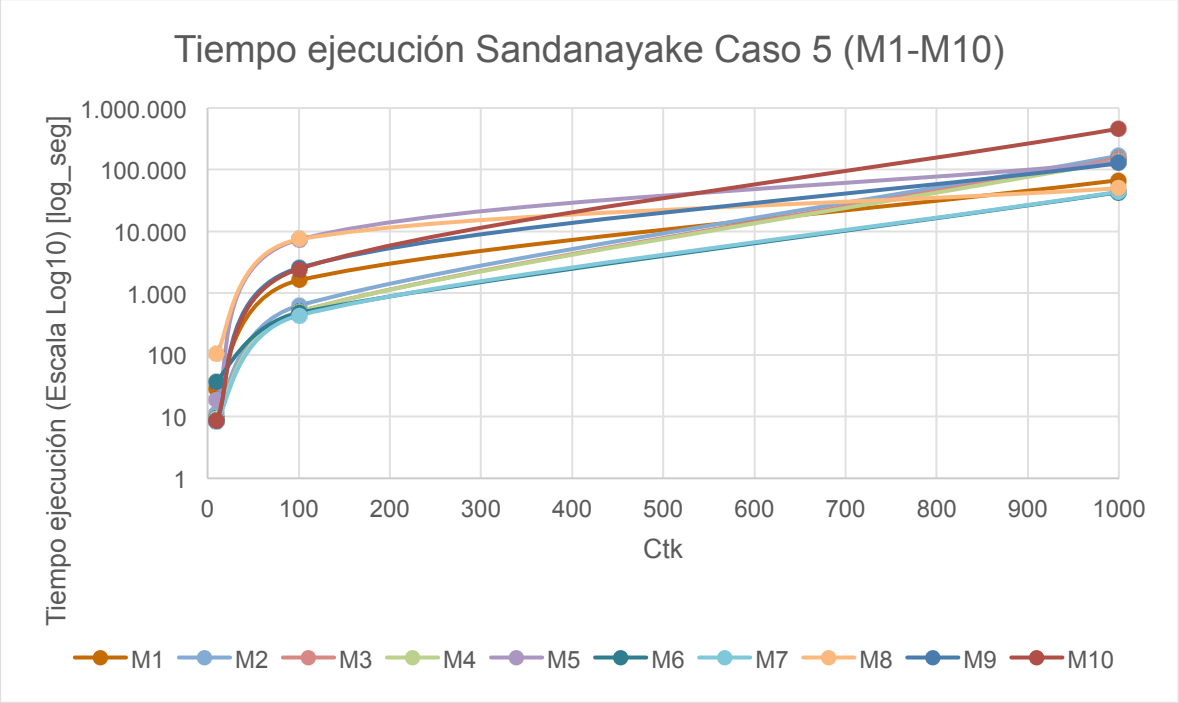


Ilustración 36: Tiempo ejecución modelo Sandanayake en función de Ctk – Caso 5

En el caso 4 y 5 se observa una situación similar. Hay un claro aumento en los tiempos de ejecución, siendo la razón principal el hecho que existe un aumento en la cantidad de caserones input al modelo de optimización (8500 y 10000 caserones, respectivamente en cada caso). Así, los números ascienden a 29.84 [seg] promedio para Ctk=10, 923.71 [seg] para Ctk=100, y 95642.45 [seg] (26.57 [hrs]) para el mayor Ctk. Lo anterior correspondiente al caso 4. El caso 5, por su parte, presenta los siguientes valores: 24.18 [seg], 2391.44 [seg], y 139016.72 (38.62 [hrs]) en la medida que incrementa la cantidad de familias (Ctk).

Todo lo anterior resume en el hecho que los tiempos asociados al menor Ctk son rápidos y del orden de segundos, para el Ctk=100 se encuentran en el orden de minutos (en promedio), mientras que para el mayor Ctk se tienen horas. Es de relevancia entonces comparar los tiempos de ejecución incurridos por el modelo de Sandanayake y los resultados obtenidos según cada Ctk en función del beneficio económico, el cual corresponde al valor maximizado.

De la Ilustración 17 a Ilustración 21 e Ilustración 32 a Ilustración 36, es posible observar la relación existente ya que en estos gráficos se presenta el comportamiento incurrido para cada Ctk en función del tiempo y el beneficio que aportan. Es posible observar que la mejora en beneficio económico no es demasiada para un Ctk alto, como se puede notar en las curvas de la Ilustración 17-Ilustración 21,las cuales tienen una pendiente baja. Esto a su vez está asociado a un mayor tiempo de ejecución.

Tabla 8: Tabla comparativa tiempos ejecución y gap - Sandanayake

	Comparación tiempo ejecución y gap promedio Sandanayake					
	Tiempo ejecución promedio [hrs]			Gap promedio [%]		
	Ctk 10	Ctk 100	Ctk 1000	Ctk 10	Ctk 100	Ctk 1000
Caso 1	0.03	2.27	252.92	7.80%	6.70%	6.40%
Caso 2	0.10	5.69	682.20	7.73%	7.21%	6.63%
Caso 3	0.15	20.88	970.62	1.92%	2.48%	2.38%
Caso 4	0.50	15.40	1594.04	16.63%	16.53%	15.69%
Caso 5	0.40	39.86	2316.95	3.83%	3.70%	3.64%

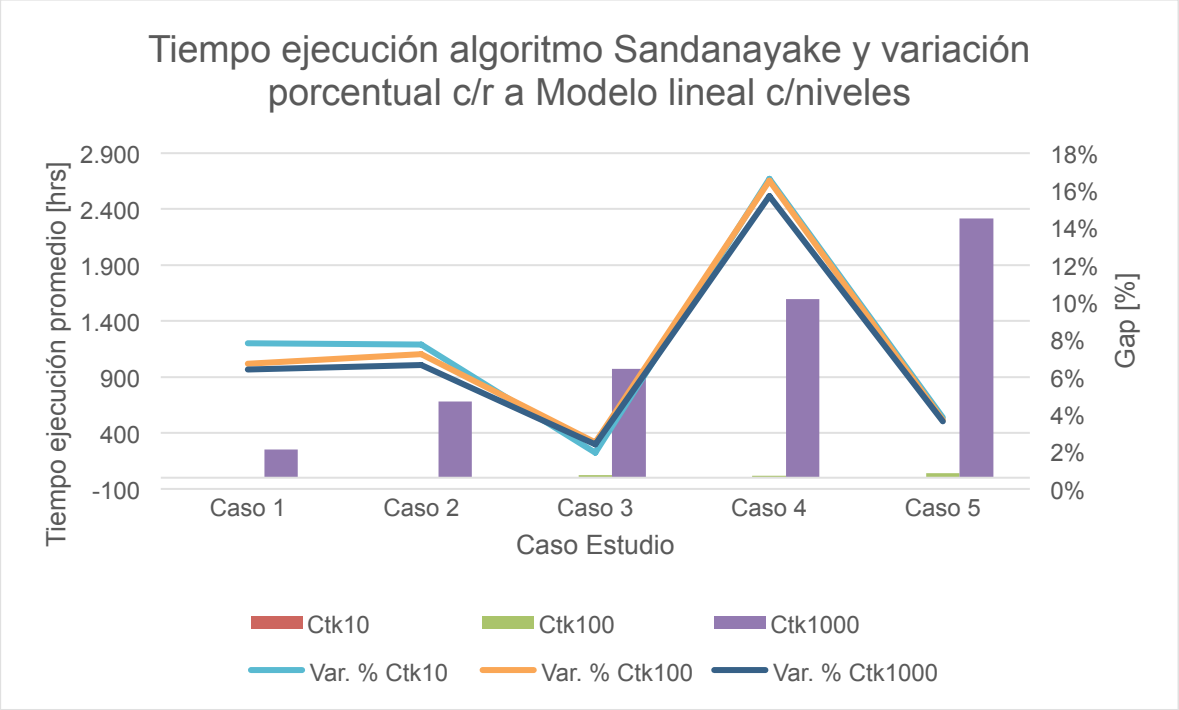


Ilustración 37: Gráfico comparativo tiempo ejecución y Gap - Sandanayake

De la Tabla 8 e Ilustración 37, es posible comparar el tiempo de ejecución incurrido para cada Ctk, y en cada caso de estudio, con el gap promedio asociado a la diferencia con el modelo lineal (óptimo). Es posible observar que para el caso de Ctk = 10 y Ctk = 100, las barras que representan el tiempo de ejecución tienen baja altura, esto es debido al bajo tiempo de ejecución incurrido. Por su parte, las líneas de variación porcentual (gap) muestran que existe poca variación entre ellas (distintos Ctk), y que su valor promedio varía en función de los resultados incurridos en cada modelo estudiado.

Por otro lado, se tienen los tiempos asociados al modelo lineal con niveles predefinidos, la cual se obtuvo a partir del código de Python y la utilización de Gurobi como solver. Los tiempos relacionados a estas corridas se presentan en la Tabla 9.

Tabla 9: Tiempo ejecución – Modelo lineal c/niveles

		Caso 1	Caso 2	Caso 3	Caso 4	Caso 5
Tiempos Ejecución Modelo lineal c/niveles [seg]	M1	31.92	57.31	81.65	497.8	1435.20
	M2	26	48.79	56.91	526.64	846.36
	M3	14.88	66.99	50.43	323.32	1175.64
	M4	32.99	62.41	75.54	126.17	1394.54
	M5	24.52	40.39	142.33	95.4	1620.92
	M6	22.7	61.77	56.12	363.44	1438.96
	M7	22.26	63.28	58.1	141.92	1362.37
	M8	36.92	81.88	206.75	519.51	836.31
	M9	45.12	39.95	231.73	444.03	1586.74
	M10	37.16	49.19	56.95	609.29	1392.75

Con el fin de comparar los tiempos incurridos por el modelo algorítmico y el modelo lineal, se presenta la tabla a continuación, en la que es posible observar las estadísticas básicas relacionadas.

Tabla 10: Estadísticas básicas tiempos ejecución modelo Sandanayake e IP c/niveles

		Estadísticas básicas tiempos ejecución [seg]			
		Promedio	Desv. Std.	Mínimo	Máximo
Caso 1	Ctk 10	1.53	1.27	0.47	4.33
	Ctk 100	136.35	125.15	25.56	361.17
	Ctk 1000	15175.30	12861.49	3273.42	43572.23
	M. Lineal	29.45	8.98	14.88	45.12
Caso 2	Ctk 10	5.36	5.20	1.36	19.30
	Ctk 100	380.68	276.34	67.60	875.84
	Ctk 1000	39760.30	29926.72	8251.54	101733.59
	M. Lineal	60.03	16.75	39.95	90.77
Caso 3	Ctk 10	9.01	4.95	2.88	17.12
	Ctk 100	1252.92	2053.27	198.50	6960.50
	Ctk 1000	58236.98	37361.00	20806.15	146980.59
	M. Lineal	101.65	67.74	50.43	231.73
Caso 4	Ctk 10	29.84	40.91	5.74	139.39
	Ctk 100	923.71	1118.95	282.37	3863.97
	Ctk 1000	95642.45	42400.70	32249.62	177123.62
	M. Lineal	364.75	187.02	95.40	609.29
Caso 5	Ctk 10	24.18	29.51	8.12	103.30
	Ctk 100	2391.44	2751.61	433.64	7405.97
	Ctk 1000	139016.78	123260.72	43108.52	461957.01
	M. Lineal	1308.98	274.81	836.31	1620.92

Es posible notar que el modelo lineal, además de asegurar el óptimo, lo hace a tiempos muy inferiores en relación al algoritmo de Sandanayake. Si bien, para un Ctk =10, Sandanayake presenta menores tiempos de ejecución, comparando para un Ctk =100, el modelo lineal entrega mejores resultados. Es de mencionar el

hecho que los tiempos de ejecución asociados al modelo lineal no incluyen el tiempo de pre-procesamiento requerido para la generación de los sets de optimización (Sección 5.1.1). Las estadísticas básicas de éstos se presentan en la Tabla 11. Básicamente, los tiempos incurridos en esta etapa no consideran parte de la optimización; sin embargo, puede ser de importancia tenerlos en consideración.

Tabla 11: Estadísticas básicas tiempos pre-procesamiento – IP c/niveles

	Estadísticas básicas tiempos pre procesamiento Modelo lineal c/niveles [seg]			
	Promedio	**Desv. Std.**	**Mínimo**	**Máximo**
Caso 1	750.00	147.25	596.56	1114.49
Caso 2	2082.10	321.77	1716.92	2683.67
Caso 3	5744.12	596.31	4871.00	6238.97
Caso 4	8137.08	297.18	7544.73	8522.91
Caso 5	14785.94	74.04	14626.79	14864.89

6.2. Modelo lineal sin niveles – Caso General

Como se presentó en secciones anteriores, el modelo lineal para la obtención de la envolvente económica con niveles predefinidos, no necesariamente corresponde al óptimo global de un cierto yacimiento. Lo anterior se tiene principalmente debido a que precisamente se está limitando la búsqueda del layout óptimo de caserones a configuraciones determinadas de losa predefinidas. Es por ello, que se propone el modelo sin niveles por medio de un modelo de optimización que incorpore una restricción de Crown Pillar a la extracción de un cierto caserón.

Para ello, se utilizó un archivo de caserones con las características de la Tabla 5, y según lo expuesto en la sección 5.2. Los resultados obtenidos se presentan a continuación.

Tabla 12: Resultados Caso Estudio General

	Ctk 10	**Ctk 100**	**Ctk 1000**	**M. lineal c/nivel**	**M. lineal libre**
N° caserones en óptimo	48	48	48	48	48
Ley promedio caserones óptimo [%]	0.65	0.65	0.66	0.69	0.70
Beneficio económico total [MUS$]	29.96	30.70	31.05	36.38	38.09

Tiempo ejecución [seg]	1.06	90.87	3608.71	5.09	238130.83
Ctk final	30	400	3,000	-	-

En el caso del modelo de lineal con niveles predefinidos el resultado se obtuvo con un pre-procesamiento de 89.59 [seg], para un resultado de gap reportado por Gurobi de 0%. Por su parte, para el modelo libre el tiempo de pre-procesamiento fue de 729.39 [seg], y se obtuvo el óptimo con un gap de 0% (reportado por Gurobi).

Es posible observar de la tabla anterior que para todos los casos estudiados se obtuvo la misma cantidad de caserones en el layout óptimo. Las diferencias radican en la elección de los caserones escogidos mediante la maximización del beneficio económico. Es posible notar el aumento en la ley media del óptimo, en el caso de los modelos lineales en relación a aquellos corridos por medio de la heurística. Como era de esperarse, el caso donde no se limita en niveles predefinidos entrega un mejor resultado, con un incremento de 1.71 [MUS$] con respecto al modelo lineal con niveles. Esto corresponde a un gap de 4.70%.

A continuación, se presentan las configuraciones de layout óptimo obtenidas del caso de estudio.

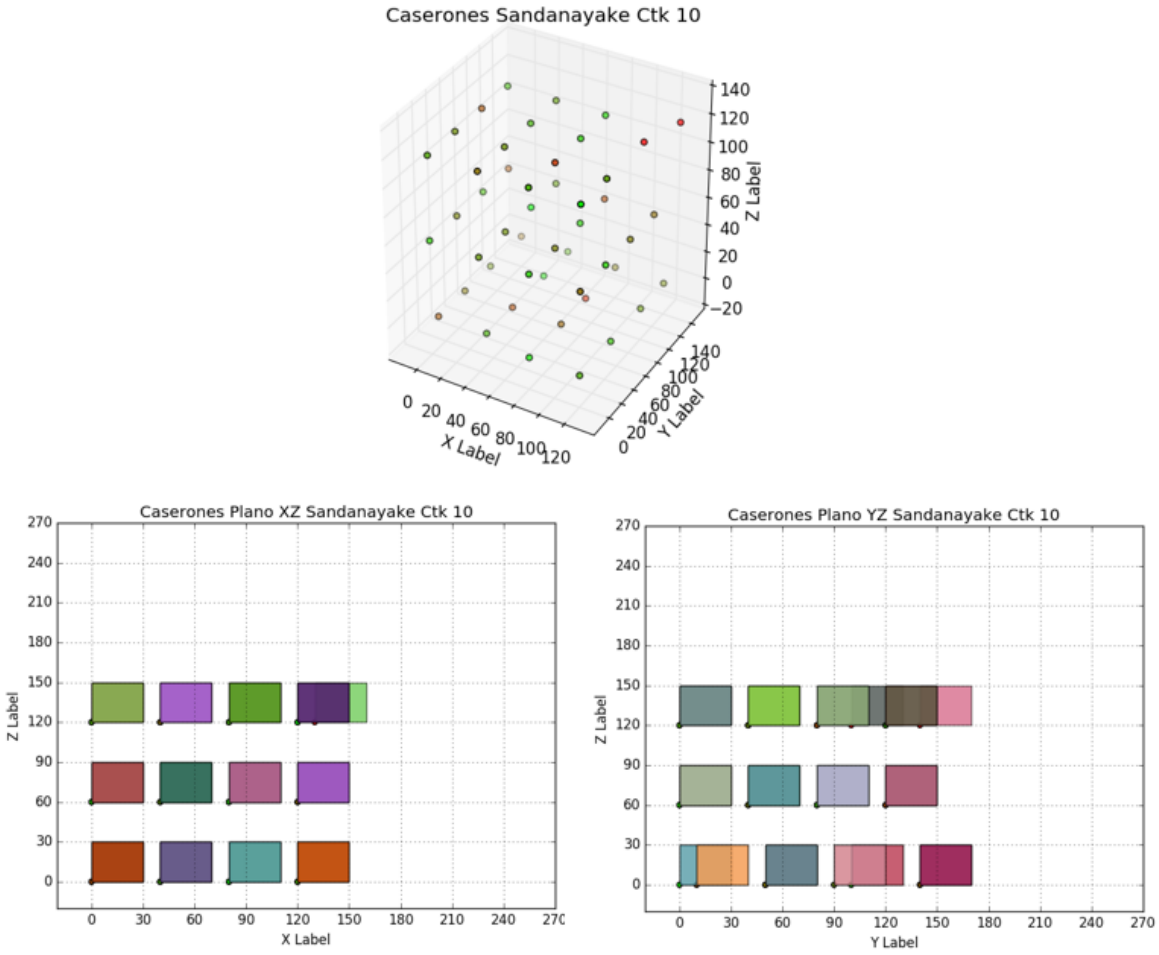


Ilustración 38: Caserones Caso Estudio General Sandanayake Ctk=10 – 3D, Plano XZ, Plano YZ

Es posible observar de la Ilustración 38, el Crown Pillar de 30 [m] entre cada nivel de caserones. Este se impuso mediante la búsqueda de todos los caserones factibles cada 30 [m] en la coordenada vertical. Según los planos XZ e YZ, pareciera que existe traslape entre caserones asociados a un no cumplimiento del pilar en XY; sin embargo, en la Ilustración 39 es posible notar que, en cada cota de caserones, efectivamente ningún caserón traslapa (y todos cumplen con la distancia mínima de 10 [m] de pilar impuesta).

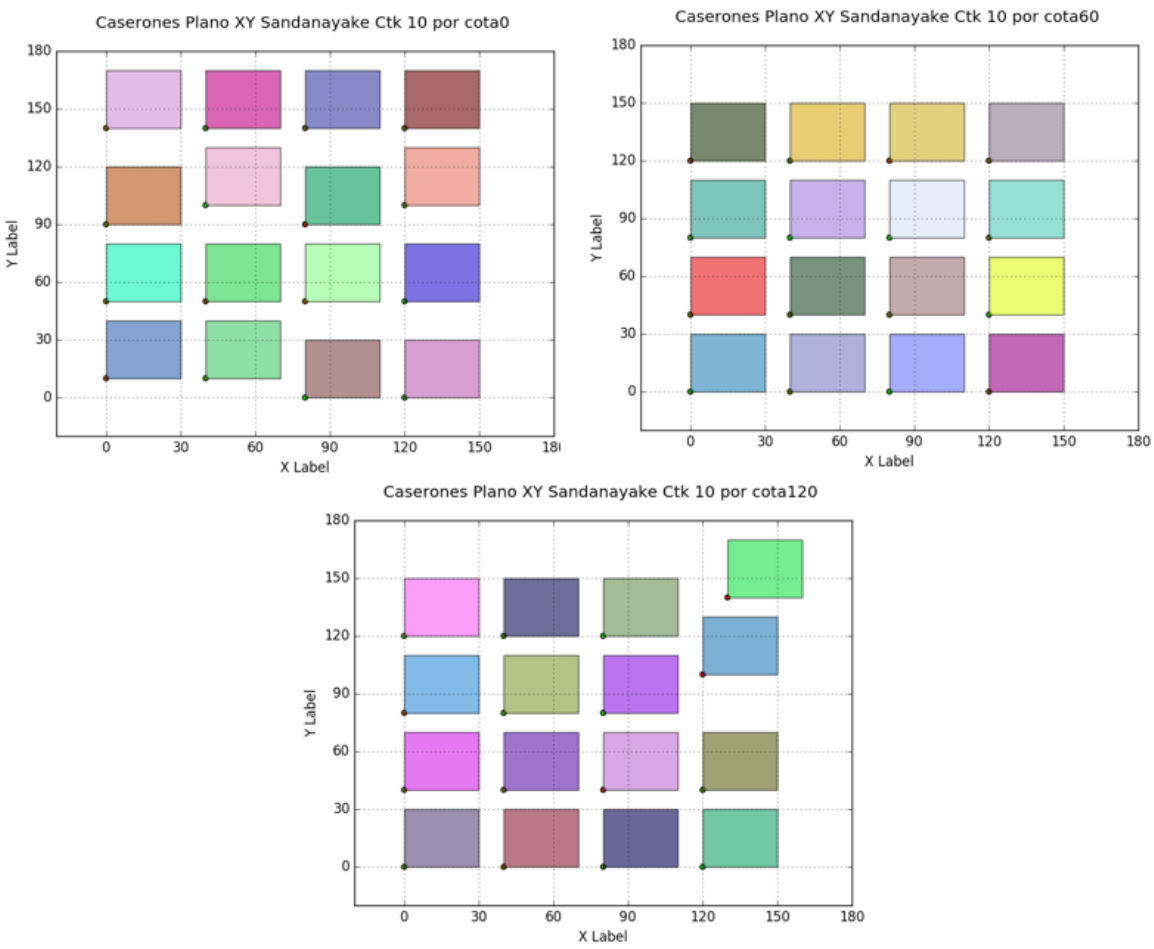


Ilustración 39: Caserones Caso Estudio Sandanayake Ctk=10 – Plano XY: Cota 0, Cota 60, Cota 120

De forma similar al caso obtenido por medio del algoritmo de Sandanayake para un límite en la cantidad de familias a generar – Ctk – de 10, se tiene que no existe traslape en el caso Ctk=100; y, por ende, se cumple con la restricción de pilares impuesta. Sin embargo, es de notar que la configuración obtenida es distinta, asociado a un beneficio mayor por 0.74 [MUS$]. Esto es posible observarlo en la Ilustración 40-Ilustración 42.

Para el caso Ctk=1000, en la Ilustración 43 e Ilustración 44 se puede observar la configuración obtenida asociada a un incremento de 0.35 [MUS$] con respecto al Ctk=100, y un incremento de 1.09 [MUS$] con respecto al Ctk=10.

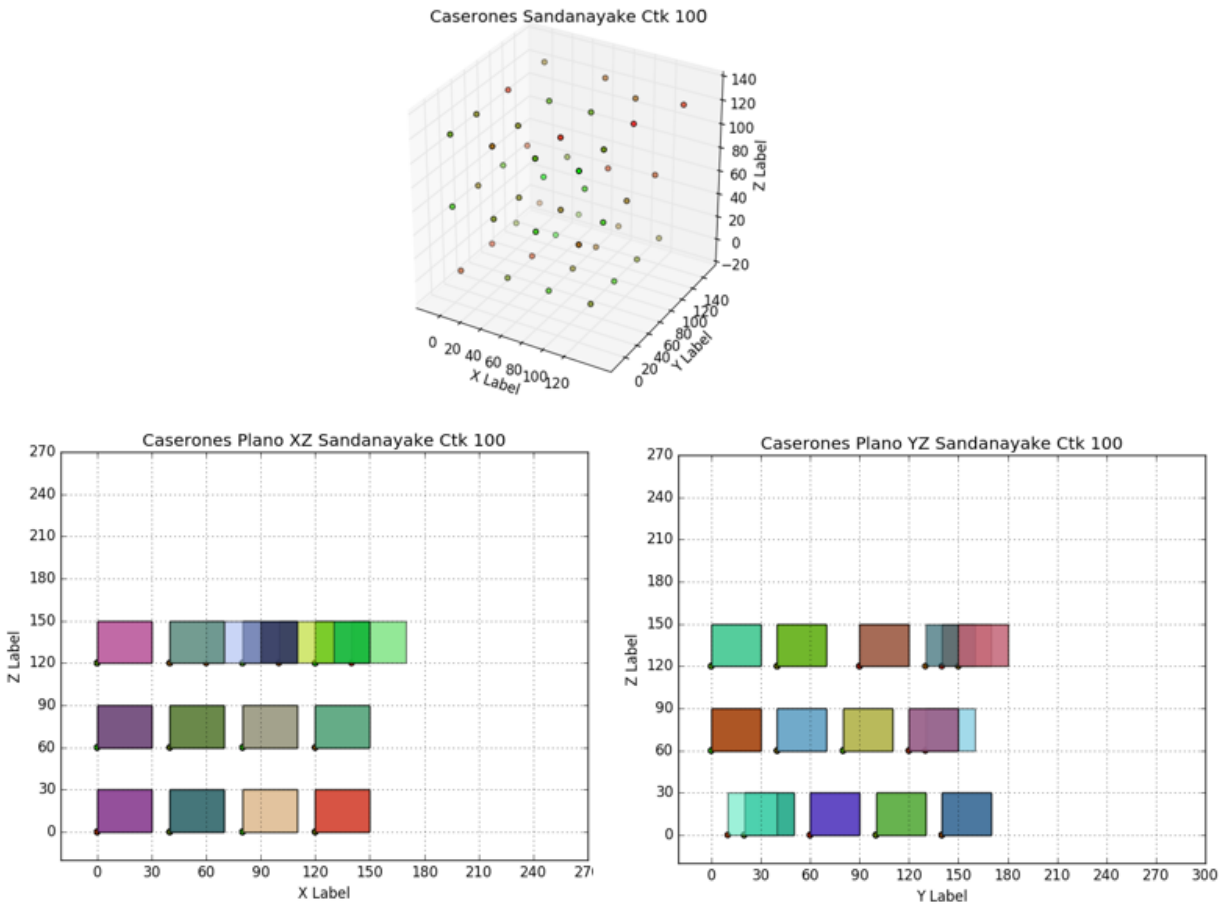


Ilustración 40: Caserones Caso Estudio General Sandanayake Ctk=100 – 3D, Plano XZ, Plano YZ

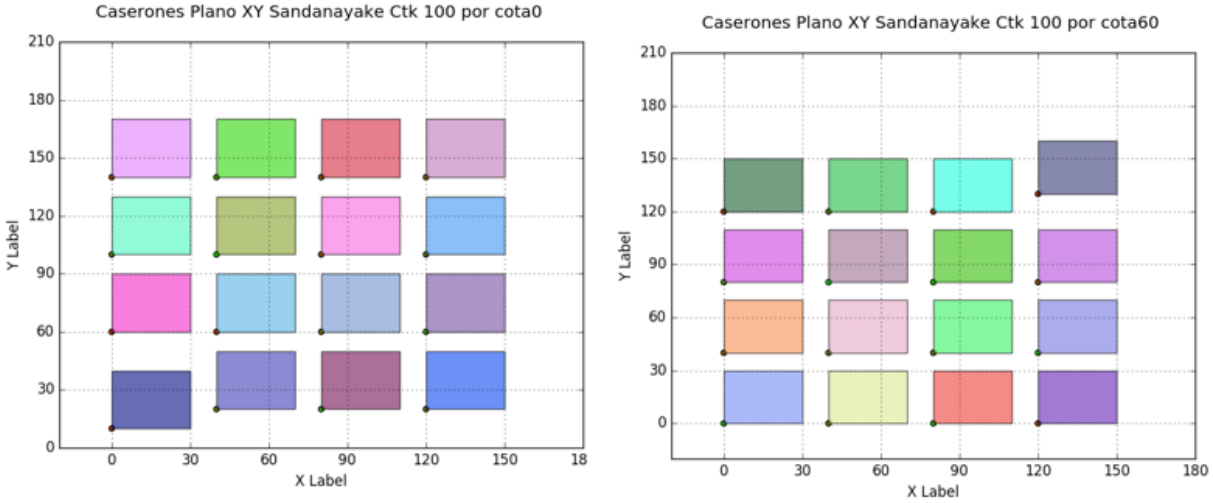


Ilustración 41: Caserones Caso Estudio Sandanayake Ctk=100 – Plano XY: Cota 0, Cota 60

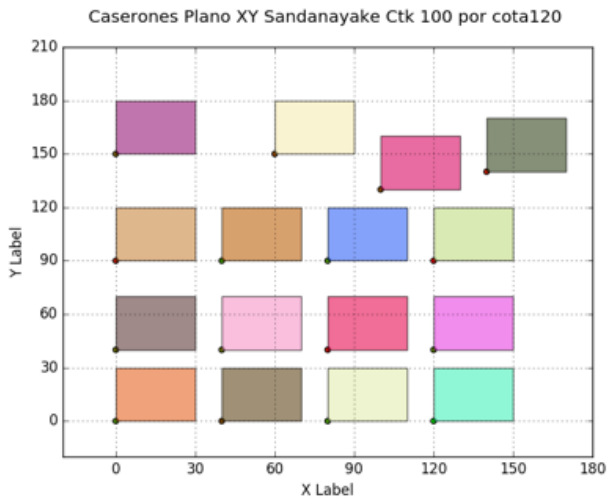


Ilustración 42: Caserones Caso Estudio Sandanayake Ctk=100 – Plano XY: Cota120

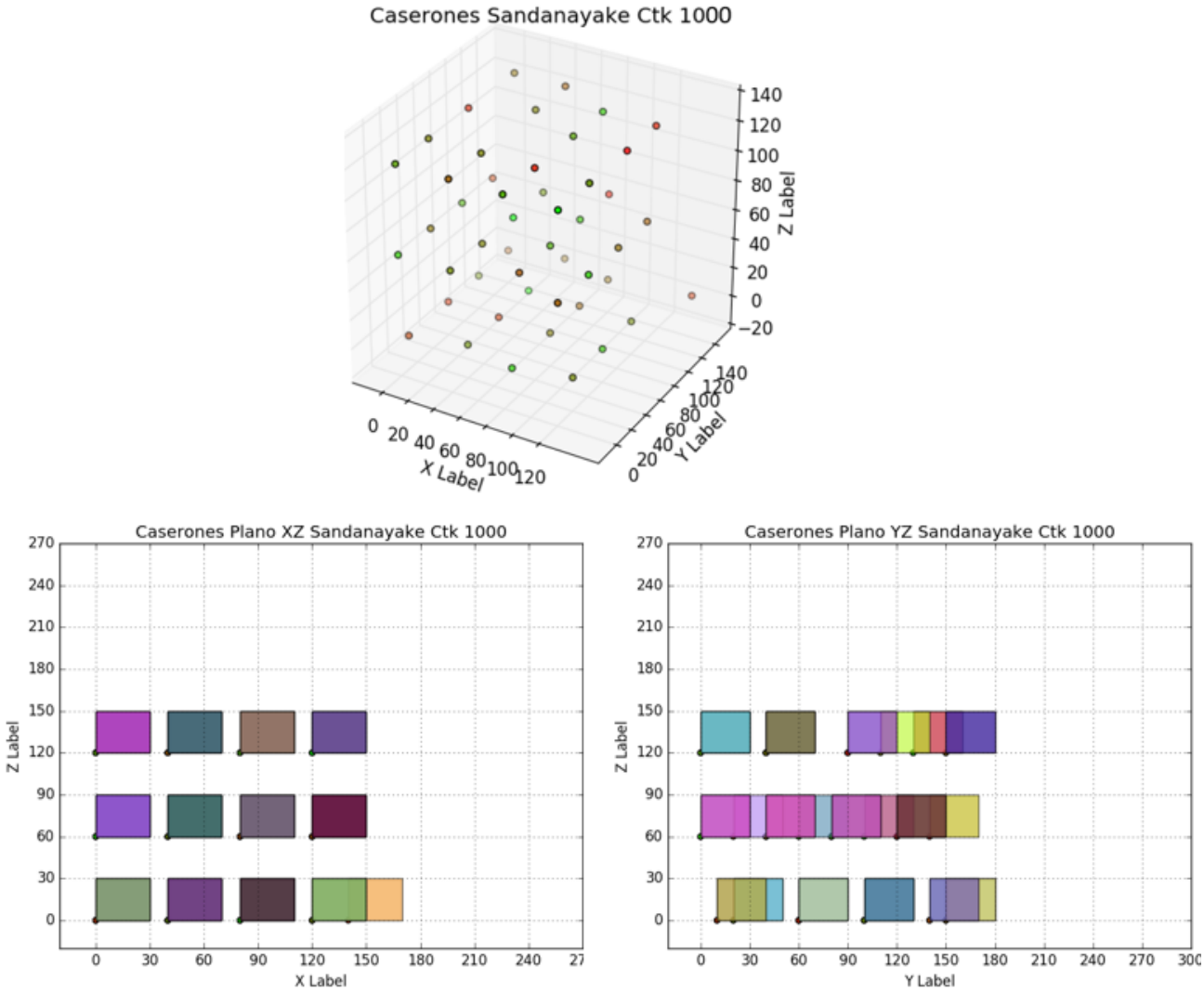


Ilustración 43: Caserones Caso Estudio General Sandanayake Ctk=1000 – 3D, Plano XZ, Plano YZ

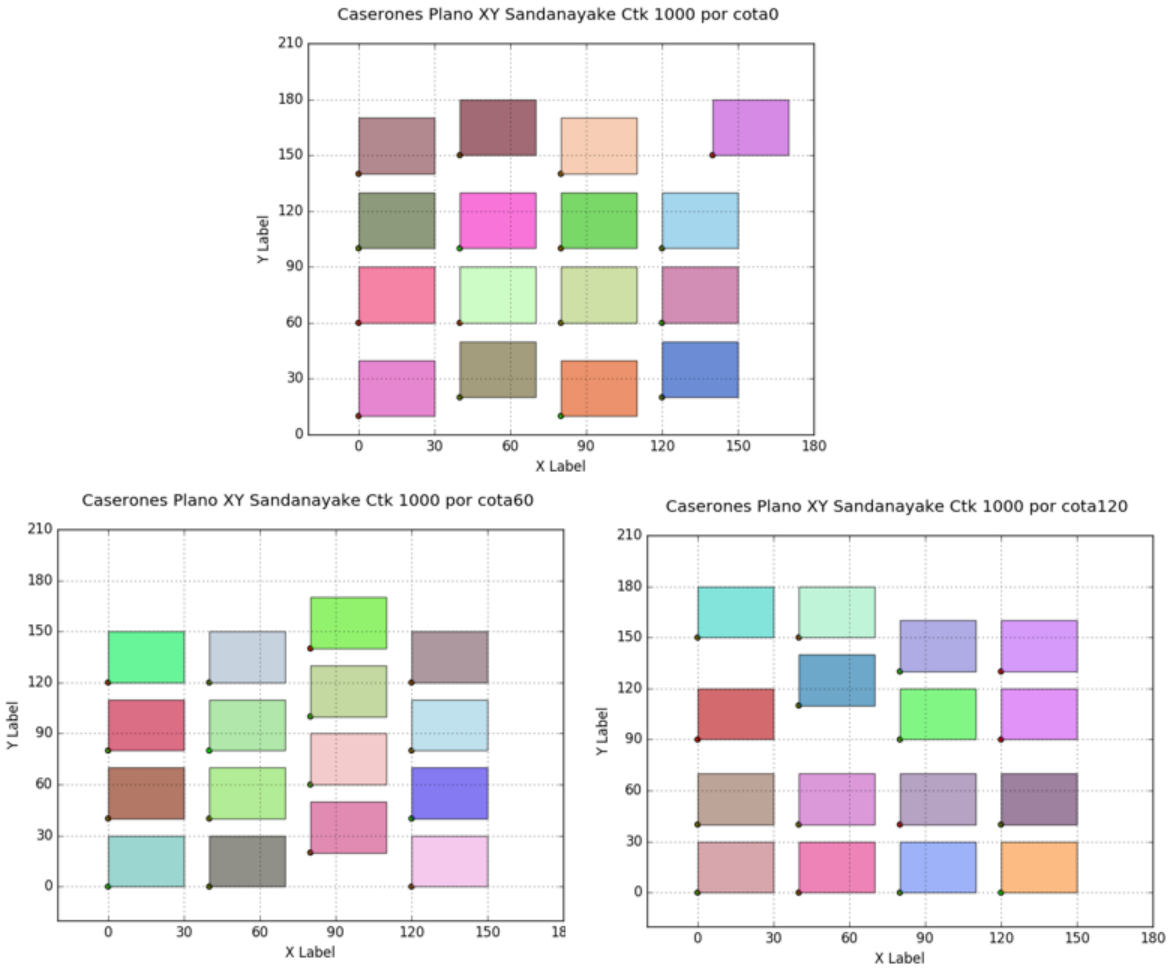


Ilustración 44: Caserones Caso Estudio Sandanayake Ctk=1000 – Plano XY: Cota 0, Cota 60, Cota 120

Por su parte, el modelo lineal con niveles predefinidos, entrega la configuración obtenida en las ilustraciones siguientes. Se observa que el layout de caserones es distinto a aquellos obtenidos con el modelo algorítmico.

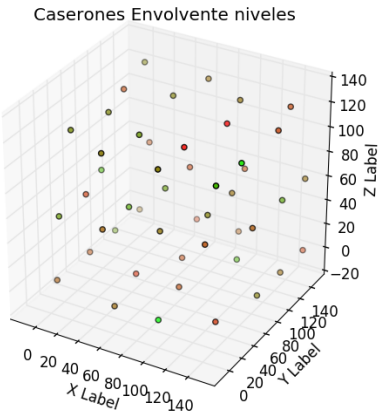


Ilustración 45: Caserones Caso Estudio General Modelo lineal Niveles – 3D

El beneficio obtenido por esta configuración corresponde a 36.38 [MUS$], una diferencia porcentual de aproximadamente 17% con respecto al resultado obtenido con Ctk=1000.

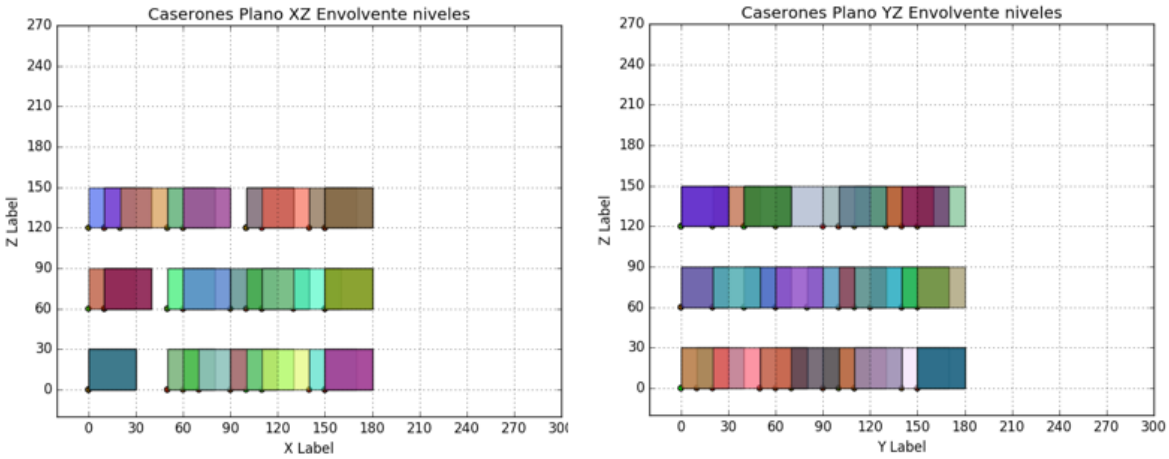


Ilustración 46: Caserones Caso Estudio General Modelo lineal Niveles – Plano XZ, Plano YZ

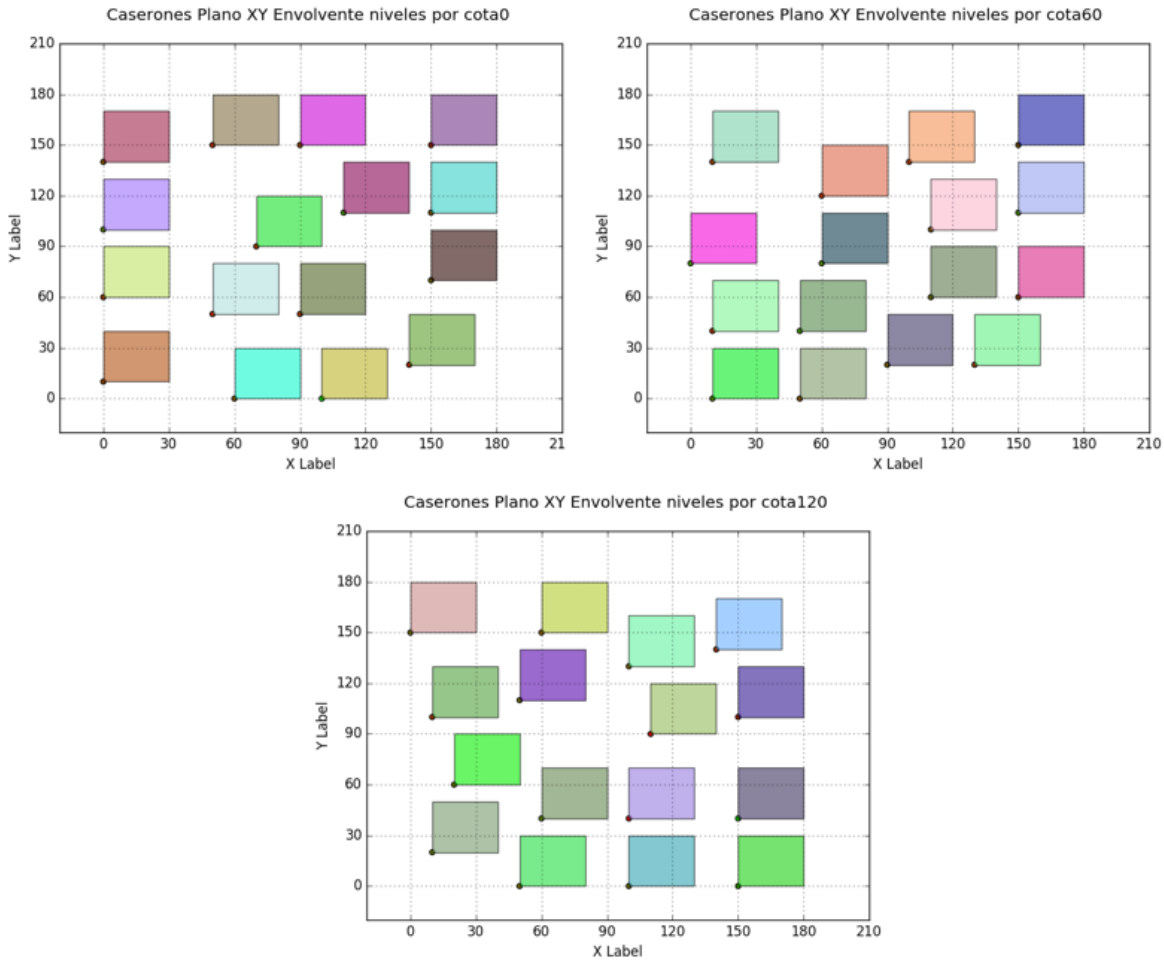


Ilustración 47: Caserones Caso Estudio Modelo lineal Niveles – Plano XY: Cota 0, Cota 60, Cota 120

Por último, se tienen los resultados obtenidos a partir del modelo lineal libre, el cual corresponde a la optimización del layout de caserones sin niveles predefinidos. Se puede notar que la optimización determinó que los niveles de caserones se encuentran en la cota 30 [m], cota 90 [m] y cota 150 [m], respetando la restricción de Crown Pillar impuesta. Esto se puede observar en la Ilustración 48.

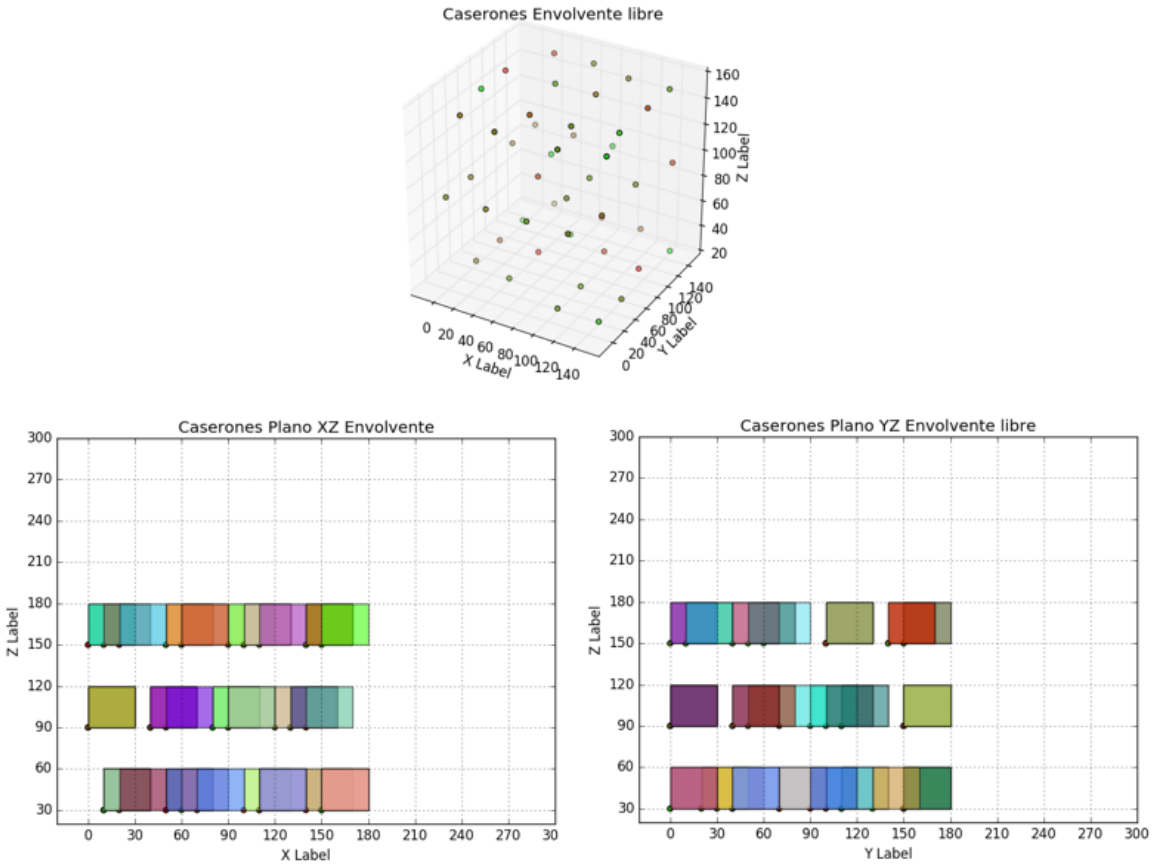


Ilustración 48: Caserones Caso Estudio General Modelo lineal libre – 3D, Plano XZ, Plano YZ

Asimismo, en la Ilustración 49, se grafican los planos XY para cada cota elegida por la optimización. Se observa el cumplimiento del ancho de los pilares (10 [m]) entre cada caserón.

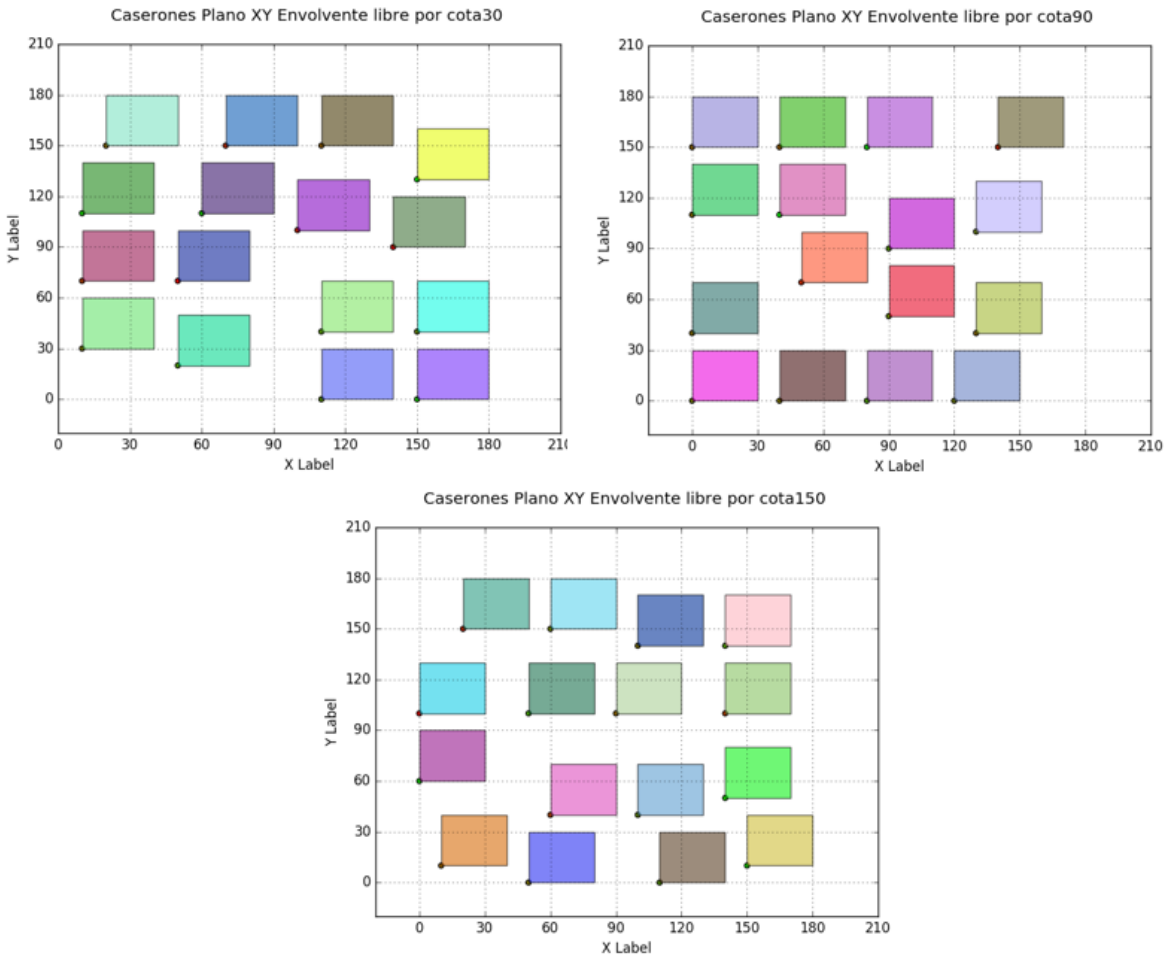


Ilustración 49: Caserones Caso Estudio Modelo lineal Niveles – Plano XY: Cota 30, Cota 90, Cota 150

Por último, en términos de tiempos incurridos por la ejecución de los modelos con niveles predefinidos (Sandanayake e IP con niveles), se tienen resultados similares a los presentados en la sección 6.1.2. En cuanto al modelo de Envolvente lineal libre, se tiene que el tiempo incurrido en la ejecución es de 238130.83 [seg] (66.15 [hrs]), superando en 65.15 [hrs] al algoritmo de Sandanayake con mayor Ctk. Sin embargo, es de reiterar el hecho que este modelo entrega el máximo global para el layout óptimo de caserones del yacimiento estudiado, ya que no se encuentra predefinido a una cierta configuración de niveles.

7. ANÁLISIS

El algoritmo de Sandanayake (2014), corresponde a una heurística potente en la obtención del layout óptimo de caserones que maximiza el beneficio. Básicamente, genera una familia de sets únicos de caserones que cumplen con la restricción de que no traslapamiento, y además cumplen con la distancia del pilar impuesto. Sin embargo, su implementación significa un gran costo computacional que se traduce en un factor tiempo. Esto se debe a que busca todas las posibles combinaciones de caserones factibles que no traslapan, y escoge posterior a ello la que maximiza el valor total de extracción. El problema aumenta considerando que, dependiendo del tipo y tamaño del yacimiento, se tendrán más o menos caserones posibles; y, por ende, la búsqueda se hace imposible. El mismo autor del modelo indica que los tiempos de encontrar solución son exponenciales, razón por la cual agrega una cota al número de soluciones o sets de caserones durante cada iteración. A esta la llamó Ctk.

Con base en los resultados presentados en la sección anterior, es posible determinar que es un modelo que permite obtener buenos resultados, pero que no asegura la obtención del óptimo global. Primero que todo, se encuentra el hecho que para Sublevel Stoping sin relleno, es necesario entregarle el archivo de caserones factibles con niveles predefinidos; por lo tanto, la solución siempre estará acotada a la configuración entregada. Es necesario aclarar que la heurística propuesta por el autor (D.S.S. Sandanayake) fue creada para yacimientos a ser explotados por medio de Sublevel Stoping con relleno, por lo que el archivo de caserones input para la optimización corresponde a los caserones factibles con niveles dispuestos en distancias equivalentes a la altura del caserón. En otras palabras, no se determinó incluir el Crown Pillar, por lo que se evaluaría una mayor cantidad de caserones.

Por otro lado, está el hecho que la cota por Ctk influye directamente en el resultado de la optimización. Para Ctk pequeños se obtiene un resultado de forma rápida, pero con altas variaciones con respecto al óptimo (considerando el óptimo como el obtenido a partir de la optimización por medio de un modelo lineal, y con los mismos parámetros de entrada). Como se observó en la sección 6.1.1 (Tabla 7), para Ctk=10, se tiene un gap promedio que varía entre 1.92% y 16.63% (con respecto al modelo lineal con niveles) según la cantidad de caserones factibles de entrada. Esto se atribuye principalmente a la forma como se obtuvo la familia de sets únicos de caserones, la cual se realiza iterando por cada caserón del modelo, y a su vez iterando según cada familia creada para verificar si existe traslape entre el caserón de la iteración y aquellos que ya están incorporados en el set. En cuanto al Ctk=100 y Ctk=1000, se tienen casos similares; sin embargo, con variaciones porcentuales menores.

En términos de diferencia entre variaciones porcentuales por caso (Caso 1 a Caso 5); los cuales varían en la cantidad de caserones (2300-10000 caserones factibles); no existen mayores diferencias según el Ctk escogido. Se observa

principalmente una disminución en la diferencia entre el óptimo del modelo de Envolvente con niveles y el valor obtenido con el Ctk de mayor valor (Ctk=1000) en cada caso. Esto es acorde a lo esperado, ya que se considera una mayor cantidad de sets únicos de caserones durante cada iteración, para así escoger la solución de mayor beneficio económico. Relacionado con lo anterior, está la cantidad de iteraciones que realiza el algoritmo heurístico para la obtención de la solución final. Se observa que en promedio son 3 iteraciones para cada caso y Ctk, pero el máximo es función de la cota impuesta. Se tiene que a menor Ctk la cantidad de iteraciones incurridas es mayor (Aproximadamente 6 iteraciones para Ctk 10, 5.8 para Ctk 100, y 4.6 para el mayor Ctk).

En términos de los tiempos de ejecución del algoritmo de Sandanayake, se tiene que éstos son función de la cantidad de caserones input a la optimización, y del valor de Ctk impuesto como cota a la cantidad de familias únicas de sets de caserones a encontrar. Esto se puede observar claramente en la Ilustración 37, donde se nota el aumento de tiempo para el Caso 5 el cual consistía en un archivo de aproximadamente 10,000 caserones.

Por su parte, el modelo lineal con niveles constituye un modelo de optimización entero en el cual se imponen las restricciones de traslape y de pilar, como se realiza en la heurística de Sandanayake, pero por medio de ecuaciones matemáticas que en conjunto con la función objetivo (maximizar beneficio) conforman un modelo de optimización a ser resuelto por un solver como Gurobi. Los parámetros de entrada corresponden al mismo archivo de caserones que se entrega en la heurística (caserones con niveles predefinidos), y el ancho correspondiente al pilar.

Los resultados obtenidos mediante el modelo lineal aseguran la optimalidad, entregando soluciones con beneficio económico igual o superior a aquel obtenido por medio del modelo heurístico. De hecho, se tiene que los tiempos incurridos por este modelo son inferiores, encontrándose bajo el promedio obtenido por el Ctk=100; y entregando resultados óptimos en segundos a minutos. En términos de cómo influye la cantidad de caserones input en los tiempos de ejecución, se tiene que a mayor cantidad de caserones factibles mayor es el tiempo generado, presentando un comportamiento exponencial. A diferencia del algoritmo de Sandanayake, existe un tiempo correspondiente al procesamiento de los datos para la generación de los sets para la optimización. Este es función también de la cantidad de caserones input; sin embargo, no es considerado dentro del tiempo de ejecución.

En relación al caso de estudio general, a diferencia de los casos anteriores, se realiza una corrida extra del modelo lineal, pero incorporando una restricción adicional a aquellas del modelo lineal con niveles. Básicamente, ésta corresponde a una restricción que limita la extracción de caserones que no cumplen con la distancia requerida por el Crown Pillar, ya que el archivo de entrada corresponde a un archivo donde en cada posición del modelo de bloques existe un caserón factible asociado. De esta forma, se activan tres restricciones del problema de

optimización que corresponden al traslape, pilar y losa. Con ello en mente, el resultado obtenido debe ser igual o superior a aquél obtenido por medio del modelo lineal con niveles.

Como se observa en la sección 6.2, con todas las corridas realizadas se obtuvo la misma cantidad de caserones en el layout óptimo, esto corresponde a 48 caserones de un total de 768 caserones factibles en el caso de niveles predefinidos, y 4,096 en el caso del modelo libre. Sin embargo, los resultados muestran que las configuraciones elegidas son distintas para cada modelo evaluado (Sandanayake, IP con niveles e IP libre). Se observa un incremento de la ley media en la medida que aumenta el Ctk del modelo heurístico, de 0.65% para el Ctk menor y 0.66% con Ctk=1000. Asimismo, se tiene para los modelos lineales, donde el caso con niveles predefinidos entregó una ley media de 0.69%, y el caso libre 0.70%. Esto es consecuencia de un mayor beneficio económico que se obtiene mediante la optimización.

En términos del beneficio económico obtenido por la heurística de Sandanayake, se tiene el máximo para el mayor Ctk (1,000), asociado a una diferencia porcentual de 3.64% con respecto al Ctk=10, y un gap de 1.14% con respecto al Ctk=100. Dado que el modelo lineal corresponde al óptimo, es relevante determinar la diferencia en beneficio obtenido por este modelo y el algoritmo de Sandanayake. Se observa que existe un aumento en el beneficio, asociado a un gap de 17.17%, lo que es coherente con las leyes medias mencionadas con anterioridad.

Para el caso lineal sin niveles predefinidos, se tiene un aumento en el beneficio económico, asociado a una diferencia porcentual de 4.70% con respecto al IP con niveles, y a un gap de 22.67% comparando con el algoritmo heurístico para Ctk=1,000. Lo anterior indica que el problema lineal de optimización planteado entrega resultados óptimos, y que efectivamente superan aquellos obtenidos por el acercamiento de Sandanayake. De hecho, la corrida libre determinó que los niveles predefinidos no corresponden a los óptimos (Cotas: 0 [m], 60 [m] y 120 [m]), si no que las cotas 30 [m], 90 [m] y 150 [m] lo son; espaciados entre sí cumpliendo con el Crown pillar impuesto en la optimización, correspondiente al doble de la altura máxima del caserón (30 [m]).

En términos de tiempo de ejecución, se puede notar que éstos incrementan notablemente, pasando de 6 segundos (Modelo lineal con niveles predefinidos) a 66 horas; aproximadamente; en el caso de Modelo lineal libre. De forma similar se tiene con el tiempo de ejecución del algoritmo de Sandanayake, para Ctk=1000, en el cual se obtuvo un tiempo de corrida de 1 hora aproximadamente. Es de esta forma relevante notar el impacto que el tiempo de corrida tiene para la obtención del resultado deseado.

8. CONCLUSIONES Y RECOMENDACIONES

A partir de la investigación realizada fue posible generar e implementar un modelo de optimización que permite la obtención del layout óptimo de caserones en minería subterránea de Sublevel Stoping sin relleno, sujeto a restricciones que permiten obtener sustento a la explotación, y entregan resultados realistas. Básicamente, se comparó el modelo generado con la heurística de Sandanayake, la que permite obtener el resultado óptimo de caserones mediante la generación de una familia de sets únicos de caserones que cumplen con las restricciones impuestas, para escoger aquél set con mayor beneficio económico.

Con el fin de comparar la eficiencia del modelo en términos computacionales y de beneficio obtenido, se generaron 150 corridas del algoritmo de Sandanayake (para tres parámetros heurísticos -Ctk- distintos), y 50 corridas con el modelo lineal con niveles. Lo anterior, asociado a distintas cantidades de caserones input. De los resultados obtenidos, se observó que el mayor Ctk estudiado, entrega los mejores resultados en términos de beneficio económico. Sin embargo, esto se encuentra asociado a un tiempo de corrida mayor. Los layouts obtenidos son todos distintos para cada Ctk y modelo analizado. Se observa un gap, con respecto al óptimo, que varía entre 1.92% y 16.63%, según la cantidad input de caserones factibles. Por su parte, en términos de diferencia porcentual entre los valores Ctk escogidos, se observa que es poco lo que mejora el aumento en la cota de sets de caserones a generar, asociado a un incremento exponencial del tiempo de ejecución.

En relación a los resultados del modelo lineal con niveles se tiene que, además de asegurar el alcance del óptimo, los tiempos de corrida son muy inferiores en relación a la heurística. Es necesario resaltar el hecho que el tiempo de ejecución es función de la cantidad de caserones a evaluar; sin embargo, se observó que la resolución de la optimización se realiza en minutos (0.5 – 22 [min], según el input de caserones). En el caso del algoritmo de Sandanayake, los tiempos van de cuestión de segundos para el Ctk menor (1.53 [seg] promedio para la menor cantidad de caserones input) a horas (39 [hrs] promedio para la mayor cantidad de caserones input). Es por lo anterior, que el modelo lineal es superior en eficiencia al modelo heurístico evaluado.

Para el caso de estudio general, se observó una diferencia porcentual de 17.17% entre los resultados del modelo lineal con niveles y el algoritmo heurístico. Asimismo, se observó un gap de 4.70% asociado al aumento en beneficio económico del modelo lineal sin niveles predefinidos. Esto, se encuentra relacionado a una configuración de niveles distinta (elegida en la optimización). En términos de los tiempos de corrida, la heurística corrió en 1 [hr] para el mayor Ctk, mientras que el IP con niveles encontró el óptimo en 6 [seg]. Sin embargo, en el modelo libre el tiempo de ejecución aumentó notablemente a 66 horas.

A partir de lo evaluado, es posible afirmar que se cumplieron los objetivos de la presente investigación, ya que fue posible la implementación y obtención de un

modelo matemático – computacional que permite optimizar el diseño en minería subterránea auto-soportada, garantizando resultados óptimos en tiempos razonables. Es de resaltar la importancia que la presente investigación lleva al diseño en minería auto-soportada, ya que no solo permite obtener resultados como los expuestos anteriormente, sino que permite realizar y analizar casos con variados parámetros económicos, y distribución de leyes en el espacio, permitiendo cuantificar de cierta forma la variabilidad económica y geológica presente en la planificación minera.

En términos de recomendaciones y trabajo futuro, se podría incorporar la dilución incurrida como un parámetro geotécnico para el diseño óptimo. Asimismo, queda propuesta la determinación del layout para caserones que varían en dimensiones en cada punto del espacio factible. Se propone también la realización de un modelo que permita incorporar los desarrollos para los accesos, de forma a priori a la determinación del minado de los caserones, así como la generación de un modelo integrado de diseño y planificación minera, similar a lo realizado por J. Little (2013) en su modelo de Planificación Integrada en SLS con relleno. Esto, sin duda, debería entregar resultados superiores a los obtenidos en la presente investigación, además de generar planes de producción realistas y óptimos.

9. BIBLIOGRAFÍA

Ataee-Pour. (2000). *A heuristic algorithm to optimise stope boundaries.* University of Wollongong Thesis Collection.

Barbaro, R., & Ramani, R. (1986). Generalized multiperiod MIP model for production scheduling and processing facilities selection and location. *Technical Papers*, 107-114.

Capees, G., Doolan, J., & Neindorf, L. (2006). Stope design considerations using rock mass classification tools at the Xstrata Zinc George Fisher Mine. 37-47.

Diakité, O. (1998). *Ore Dilution in Sublevel Stoping.* Montreal: Msc. Thesis McGill University.

Fernandez, G., Fuentes, C., & Alvarado, S. (2006). Clasificación geoténica para fortificación y dimensionamiento máximo de cámaras de explotación mina El Peñon-Medirian Gold. *XI Congreso Geológico Chileno* (págs. 39-42). Antofagasta-II Región, Chile: Actas Geología Aplicada.

Henning, J. G., & Mitri, H. S. (2007). Numerical modelling of ore dilution in blasthole stoping. *International Journal of Rock Mechanics and Mining Sciences*, 692-703.

Little, J., Knights, P., & Topal, E. (2013). Integrated optimization of underground mine design and scheduling. *The Journal of the Southern African Institute of Mining and Metallurgy*, 775-785.

Milne, D. (1997). *Underground design and deformation based on surface geometry.* PhD. Thesis, University of British Columbia.

Nehring, M., Topal, E., Kizil, M., & Knights, P. (2012). Integrated short-and medium-term underground mine production scheduling. *The Journal of the Southern African Institute of Mining and Metallurgy*, 365-378.

Pakalnis, R., Poulin, R., & Hadjigeorgiou, J. (1995). Quantifying the cost of dilution in underground mines. *Mining Engineer*, 1136-1141.

Ramos, A., Sánchez, P., Ferrer, J., Barquín, J., & Linares, P. (2010). *Modelos matemáticos de optimización.* Madrid, España: Universidad Pontificia ICAI ICADE Comillas Madrid.

Sandanayake, D. (2014). *Stope boundary optimisation in underground mining based on a heuristic approach.* Australia: PhD Thesis, Western Australian School of Mines.

Sandanayake, D., Topal, E., & Asad, M. (2015). A heuristic approach to optimal design of an underground mine stope layout. *Applied Soft Computing*, 595-603.

Sandanayake, D., Topal, E., & Asad, M. (2015). Designing an optimal stope layout for underground mining based on a heuristic algorithm. *International Journal of Mining Science and Technology*, 767-772.

Scoble, M., & Moss, A. (1994). Dilution in underground bulk mining: Implications for production management, mineral resource evaluation II, methods and case histories. *Geological Society Special Publication*, 95-108.

Sens, J. (2011). *Stope mine design optimisation using various algorithms for the randgold kibali project.* Msc. Thesis Delft University of Technology.

Topal, E. (2008). Early and Late start algorithms to improve the solution time for long term underground mine production scheduling. *The Journal of The Southern African Institute of Mining and Metallurgy*, 99-107.

Villaescusa, E. (2014). *Geotechnical Design for Sublevel Open Stoping.* CRC Press Taylor & Francis Group.

10. ANEXOS

10.1. Anexo A: Código Python

Como se explicó en la sección 4.1, el código implementado en Python se encarga de la generación de los caserones factibles para cada posición del modelo de bloques. Primero realiza la lectura del modelo de bloques entregado en formato csv, así como los parámetros de entrada requeridos, entre los que se encuentran: recuperación metalúrgica, costo de venta, costo mina, costo planta, precio del metal, dimensiones del bloque en cada eje, y cantidad de bloques mínimos y máximos para la agregación y generación de caserones fijos en cada punto. Es de resaltar que el modelo de bloques entregado debe contener los siguientes parámetros: cenx (centroide x), ceny (centroide y), cenz (centroide z), cut (ley media) y ton (tonelaje); respectivamente.

Con lo anterior como entrada para el modelo, el código funcional se encuentra dividido en una clase principal llamada *Yacimiento,* y dos clases hijas llamadas *YacimientoBloques* y *YacimientoCaserones.* Se realizó la distinción entre estas últimas de manera que en la primera se trabaje con los bloques obtenidos del modelo de bloques entregado. Dado que éste posee datos desordenados; i.e. los bloques en el archivo no se encuentran bajo algún tipo de orden por coordenadas; es requerido primero ordenar lexicográficamente las filas para así poder trabajar con bloques contiguos, lo cual es útil para la generación de caserones por agregación. Para ello, se creó la función *make_sorted* que entrega una lista ordenada de filas/bloques.

Por otro lado, se tiene la función *generarcaserones_fijos* la cual permite la agregación de bloques para la creación de caserones, basado en el total de bloques máximos por cada eje. Se encuentra basado en la tesis de D.S.S. Sandanayake, como se ha explicado con anterioridad. De forma similar se tiene con la función *generarcaserones_variables* la cual realiza lo mismo que la anterior; sin embargo, considera la creación de caserones de variado tamaño para cada fila del modelo de bloques, sujeto a la cantidad de bloques mínimos y máximo impuestos como parámetros de entrada. En otras palabras, para cada fila/bloque del modelo de bloques genera el total de posibles caserones de distinto tamaño. En ambos casos, se genera un archivo csv que contiene los parámetros como centroide, tonelaje, ley y valor.

Como se mencionó en la sección 4.1.2, el algoritmo de Sandanayake (2014) incorpora los pilares durante la generación de los caserones fijos o variables al establecer que la búsqueda en el eje vertical (“z”) se realiza con un incremento equivalente a la altura máxima de los caserones. Esto, se traduce en un k=k+nz, en lugar de k=k+1, durante la agregación de bloques, razón por la cual la solución es sólo óptima para esa configuración de Crown Pillar.

En relación con la segunda clase hija *YacimientoCaserones*, se tiene que esta trabaja con los caserones. Una vez generados los posibles caserones con las funciones mencionadas anteriormente, utiliza la función *caseronespositivos* para seleccionar sólo aquellos con beneficio mayor a cero, y genera un archivo csv que contiene los parámetros como centroide, tonelaje, ley y valor. A continuación, se presenta un esquema para resumir el código en cuestión.

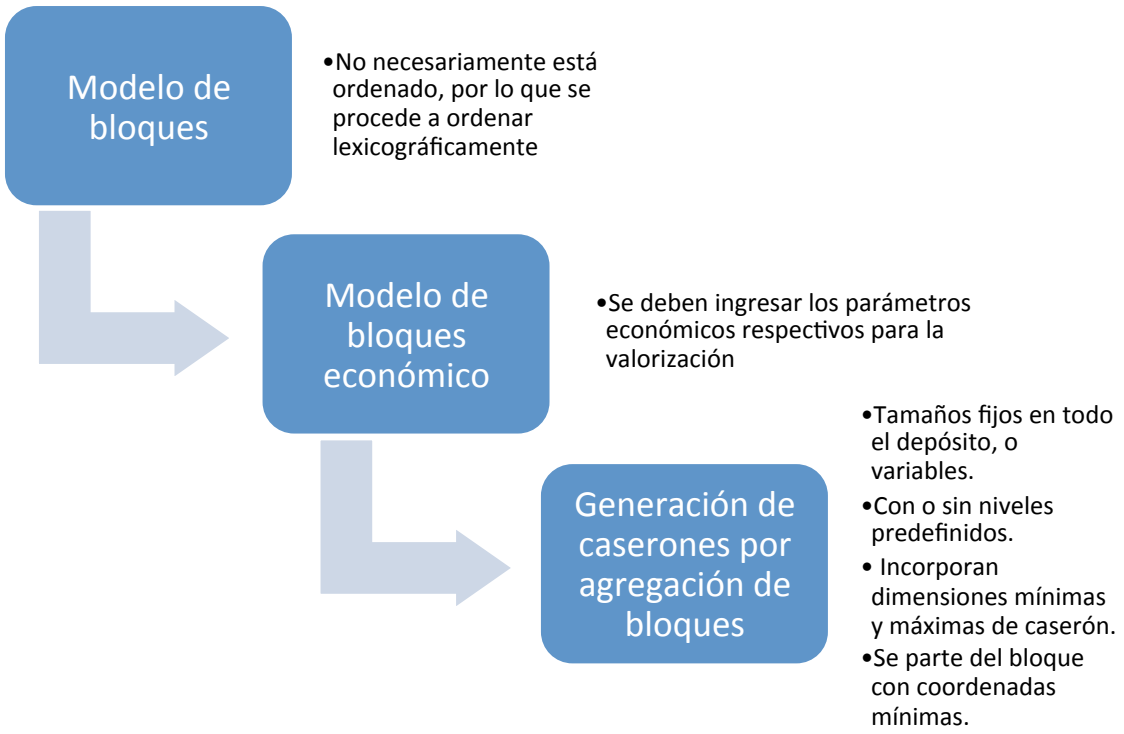


Ilustración 50: Esquema general generación caserones

10.2. Anexo B: Código Java - Generación familia de sets caserones

El código Java, como se explicó en la sección 4.1.2, se encarga de la generación de familias de sets únicos de caserones que cumplen la condición de no traslapamiento y pilares. La clase *Main.java* parte cargando el archivo csv de caserones fijos o variables obtenido de Python, y que contenga como columnas: x, y, z, sizex, sizey, sizez, cut, ton y valor; respectivamente; donde las primeras corresponden a los centroides en cada dirección y las columnas asociadas a "size" equivalen al tamaño de cada caserón en metros. Mediante la clase implementada *StopeHelper.java* se crea un arreglo que permita guardar los datos del archivo de entrada con 9 columnas, lo cual es punto de partida para la obtención de la familia de sets de caserones.

Como se observa en el diagrama de flujo del algoritmo de Sandanayake, 2014, para la obtención de la familia de sets de caserones que no traslapan (Ilustración 51), el input principal corresponde al conjunto o set *gamma_s*, el cual en el algoritmo implementado en Java corresponde al arreglo de conjunto creado con la clase *StopeHelper.java.* Con el fin de modelar el diagrama de flujo, se creó la clase *StopeSet* la cual retorna el óptimo local para el parámetro *gamma_s* de entrada y un entero Ctk, la cual es la cota máxima del tamaño de la solución intermedia que entrega la heurística de Sandanayake. El último, utiliza ese valor para acotar la solución y nunca tener más allá de Ctk soluciones diferentes. En cada iteración la familia que se genera se construye a partir de la familia anterior, y nunca tendrá más allá de Ctk elementos. Con esto, se pretende evitar que el árbol de búsqueda crezca de forma exponencial en cada nivel de recursión, quedándose con las Ctk mejores soluciones. En otras palabras, el árbol de búsqueda en lugar de ir creciendo de forma exponencial, crece hasta la cota impuesta.

El diagrama de flujo se modela en el archivo *Main.java* complementado con la implementación de las clases *FamilyStopeSet.java* y *StopeSet.java,* donde la primera permite crear un conjunto vacío que corresponde a la familia que contiene los distintos y únicos conjuntos de caserones, los cuales se generan según *StopeSet.* A su vez, *StopeSet* es generado según la clase *Stope.java* en la cual se encuentran todos los métodos/funciones que identifican y permiten operar sobre caserones. En particular, en este se encuentra la función boolean *overlaps* la que indica si dos caserones se traslapan o no.

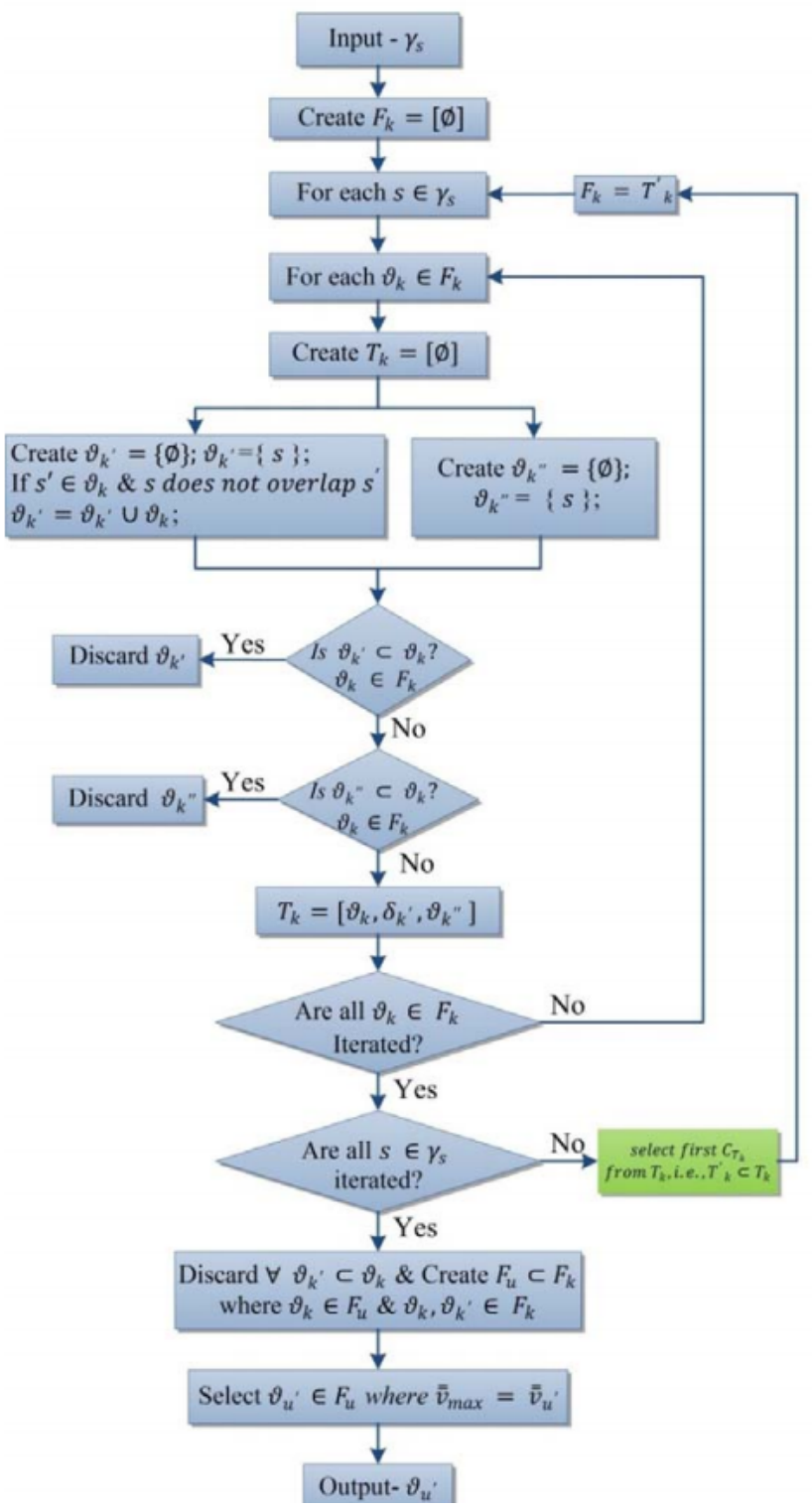


Ilustración 51: Diagrama de flujo familia sets caserones, Sandanayake 2014

Es de recalcar que esta condición se evalúa entre caserones modelados como paralelepípedos, y no entre bloques pertenecientes al caserón. La condición de traslape que se evalúa es:

$$min_x^A < min_x^B \ \& \ max_x^A > max_x^B$$
$$\&$$
$$min_y^A < min_y^B \ \& \ max_y^A > max_y^B$$
$$\&$$
$$min_z^A < min_z^B \ \& \ max_z^A > max_z^B$$

Ecuación 11: Ecuaciones traslape caserones

Donde, se comparan los valores mínimos y máximos en cada eje entre dos caserones A y B.

En relación a la clase *StopeSet* implementada se tiene que esta corresponde al conjunto de caserones viables que no traslapan entre sí. Con el fin de verificar que dos soluciones factibles no son iguales, se verifica en esta clase si es que la lista de caserones que define cada conjunto son iguales o no. Esto se realiza verificando la cantidad de caserones que contiene cada lista. Si el valor es igual, se verifica si es que cada uno de los caserones, en orden, es igual que el otro caserón de la otra lista.

La clase *FamilyStopeSet* define las funciones que permiten operar sobre una lista de conjuntos de caserones (*StopeSets*). En particular, se tiene la función *removeSetsContainedInOtherSets* la cual permite remover conjuntos de caserones que ya estuvieran contenidos en otros elementos dentro de la misma familia, de manera que ésta sea única.

Por otro lado, se tiene la función *getMostValuableStopeSet* la cual busca y retorna la solución de mayor valor económico sobre cada uno de los candidatos almacenados en la familia. A su vez, la función *sortAndKeepAtMost* ordena la lista de conjuntos de caserones de manera descendente, según la cota impuesta Ctk, eliminando todas aquellas soluciones que se encuentren fuera del rango.

Con todo lo anterior, el algoritmo sigue el diagrama de flujo expuesto anteriormente para la selección de la familia con mayor valor económico total. A continuación, se identifican las notaciones principales del diagrama de flujo:

- Gamma_s: Archivo de ingreso con caserones posibles.
- Fk: Conjunto o familia que contiene todos los posibles sets únicos de caserones.
- s: Caserón perteneciente al conjunto Gamma_s.

10.3. Anexo C: Código Python - Procesamiento de datos optimización

Con el fin de obtener los sets requeridos por los modelos de optimización se creó el archivo *parametrosopt.py* el que permite el procesamiento de los datos para ser utilizados como listas de Python. Básicamente, a partir del archivo de caserones para cada posición del modelo de bloques (obtenido según la sección 4.1.1), se buscan los datos que cumplan ciertas condiciones según se requiera en cada modelo.

Primero, se encuentra la función *obtenercolumna(archivoentrada, índice)* la cual toma como parámetros input un archivo de entrada (archivo de caserones variables con 9 columnas) y un índice. El último corresponde al índice de la columna que se desea obtener en forma de lista. Es decir, al llamar la función *obtenercolumna('caseronesvariables.csv',8)* se crea una lista de Python correspondiente a la columna 9 (recordar que índices en Python parten en cero) del archivo '*caseronesvariables.csv';* es decir, se crea una lista donde se tiene el valor económico de cada caserón.

Por otro lado, se encuentra la función *buscar_colisiones(archivoentrada)* la cual toma como parámetro de entrada un archivo de 9 columnas, como se explicó con anterioridad. Básicamente, esta función entrega una lista de listas, donde cada sublista corresponde a los caserones que traslapan con el caserón de la posición de la sublista. Para ello, se comparan las filas del archivo de entrada; correspondiente a los caserones; y se compara cada una para ver si existe traslape. El último se verifica según la Ecuación 11: Ecuaciones traslape caserones.

De forma similar, se tiene la función *obtenernivel(archivoentrada,pilar)* la cual permite obtener el set (o lista de listas) de caserones que no comparten niveles de extracción en común, y que se encuentren dentro de una vecindad acotada por el tamaño máximo de caserón y la altura del pilar (Ecuación 12).

$$Vecindad\ Niveles\ Extracción = Altura\ máxima_{caserón} + Altura\ Crown\ Pillar$$

Ecuación 12: Vecindad para búsqueda de set de niveles de extracción

Es por lo anterior, que la función toma como parámetro de entrada el archivo de caserones y un valor numérico definido como *pilar*, que corresponde a la altura del Crown pillar que se desea incorporar.

Por otro lado, se encuentra la función *casmismonivel(archivoentrada)* el cual entrega el set de caserones del archivo de entrada, que cumplen la condición de encontrarse en el mismo nivel de extracción dado por la coordenada z.

Por último, en relación a los sets de caserones requeridos para la optimización, se tiene la función *obtenerpilar(archivoentrada,pilar).* Esta se encarga de obtener el

set de caserones que no cumplen con la distancia establecida por el valor numérico *pilar* entregado por el usuario, y correspondiente al valor del pilar a ser dejado entre caserones del mismo nivel. Para ello, debido a que el archivo de entrada posee las coordenadas del origen de cada caserón, se detecta primero si existe un pilar entre dos caserones en evaluación según la Ecuación 3 modificada según las coordenadas entregadas. Posterior a ello, se encarga de filtrar los caserones que no cumplen la condición anterior, y los incorpora a la lista. Es de resaltar que la búsqueda asociada a estos caserones corresponde a una vecindad móvil con el fin de evitar que caserones que se encuentren alejados entre sí, pero no cumplan en algún eje la distancia correspondiente al pilar, sean etiquetados.

Asimismo, el programa *parametrosopt.py* se encarga de procesar la solución obtenida del modelo de optimización, mediante la función *cruzarsolucionenvolvente(archivosolucion, archivocaserones, archivosalida).* Básicamente, esta recibe como input el archivo de solución del problema de optimización, y por medio de las variables de decisión correspondientes, crea un nuevo archivo de salida el cual posee el mismo formato que el archivo de caserones entregado inicialmente. Sin embargo, este nuevo archivo se encuentra filtrado según la variable de decisión; y, por ende, es un subconjunto del archivo de caserones entregado.

10.4. Anexo D: Parámetros económicos – Casos Estudio

Tabla 13: Parámetros económicos – Casos Estudio

Parámetros Económicos	
Precio Venta Cobre [US$/lb]	3
Costo Mina [US$/ton]	8
Costo Planta [US$/ton]	12
Costo Venta [US$/lb]	0.5
Recuperación [%]	80%