Adquisición y procesamiento de información en sistemas automáticos de optimización mediante simulación[*].

Rubén Artime[1], Pilar L González Torre[2], Adenso Díaz Fernández[3]
[1] Dpto. de Sistemas de Información (Goalsystems, C/ Julio Camba, 1, 28028 Madrid, ruben@goalsystems.com)
[2] Profesor Asociado (ETSII Gijón. Campus de Viesques. 33204 Gijón, pilargt@etsiig.uniovi.es)
[3] Profesor Titular (ETSII Gijón. Campus de Viesques. 33204 Gijón, adenso@etsiig.uniovi.es)

RESUMEN

Cuando se diseña un modelo de simulación donde los datos de entrada son de gran volumen, es necesario definir un preprocesado de la información que la depure, y haga más fiable y rápido su procesamiento posterior. El presente artículo aborda este problema dentro de un proyecto que analiza datos geométricos, en concreto, relativos al transporte dentro de un entorno de producción minera, previamente a la optimización de políticas de gestión de la red vía simulación. Este preprocesamiento se realiza en varios pasos, basados en metodologías de teoría de grafos. El sistema se ha validado en la empresa minera Hunosa.

1. Introducción.

La calidad de la información de partida en el proceso de análisis mediante simulación es un aspecto que muchas veces se supone a la hora de iniciar un proyecto y que puede llevar al traste muchas horas de trabajo si finalmente no es suficientemente fiable [1]. Este caso es especialmente grave si provoca resultados finales incorrectos pero no detectados debido a que la mayor parte de ocasiones las verificaciones se realizan sobre el trabajo propio y no sobre los datos de partida. Cuando la cantidad de información es especialmente grande (como ocurre en los proyectos que trabajan con datos geométricos/geográficos, donde la información de partida puede ocupar cientos de megabytes y por lo tanto resulta impensable una verificación manual de los datos iniciales) o cuando el proceso es de tipo on-line, resulta natural tratar de automatizar de alguna manera esta verificación inicial.

En el ámbito de la minería subterránea, una de las mayores dificultades de planificación es la de programar los movimientos del transporte de material [2]: por las características propias del entorno donde se produce la extracción (falta de espacio), una mala programación del transporte da lugar a que la instalación tenga que paralizar su actividad si no se evacua el material a tiempo y el espacio intermedio de almacenamiento dispuesto en el punto de producción se ve saturado.

La simulación se ha constituido como una de las metodologías más utilizadas en la optimización de sistemas complejos. Su capacidad para dar rápida respuesta a cuestiones de tipo "what-if" permite que, si se incorpora dentro de un sistema inteligentemente diseñado para analizar distintas alternativas eficientes, pueda dar respuesta a cuestiones de tipo estratégico difícilmente modelables analíticamente [3,4]. En este trabajo, se trata pues de desarrollar una herramienta para

[*] Investigación financiada a través del proyecto europeo CECA-7220-PR/034.

la simulación y optimización de redes de transporte de material en el interior de las minas, que permita una evacuación eficiente del material.

A la vista de la naturaleza dinámica de la estructura de transporte en las minas, donde por el avance de las zonas de extracción continuamente se están modificando los trazados ferroviarios, previamente al diseño del sistema de optimización se hace necesario diseñar un sistema que permita a aquél disponer de información precisa sobre los trazados reales y las situaciones de los puntos de producción, a partir de la cual poder encontrar las mejores políticas de explotación. En otro caso, el tratamiento manual necesario para actualización de datos sería muy costoso.

En este caso, la información básica de partida (además de las relativas a las características propias de cada zona de trabajo, como puede ser la curva de producción o las capacidades de almacenaje) la constituyen los tramos de las vías que forman una red que conecta las distintas zonas de producción de material, con las zonas desde donde el material es evacuado. De este modo el simulador podrá representar los movimientos de los vagones entre cualesquiera puntos de modo realista.

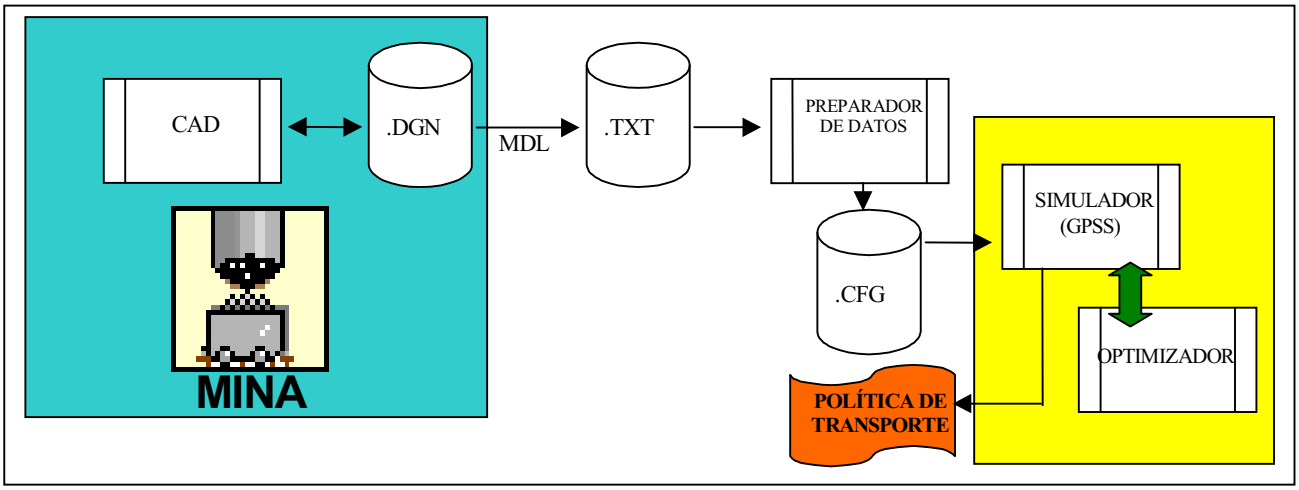

Figura 1. La preparación de los datos como interface entre el sistema real y el simulador

En este trabajo se presentan los algoritmos diseñados para el sistema que permite captar de modo automático (adquisición, verificación, corrección y construcción), a partir del sistema topográfico de información geográfica interno de la mina (por lo demás siempre con errores topológicos que habrá que detectar y corregir para poder automatizar el proceso), toda la información necesaria para el posterior funcionamiento del optimizador. No son abordadas en este trabajo, sin embargo, las cuestiones relacionadas con el diseño del simulador y su optimización, los cuales necesitan de estos datos básicos para sus cálculos (figura 1).

2. Pasos en la adquisición y preparación de los datos.

El sistema de adquisición y depuración de la información de diseño que aquí se propone consta de una serie de pasos secuenciales que se detallan a continuación (figura 2).

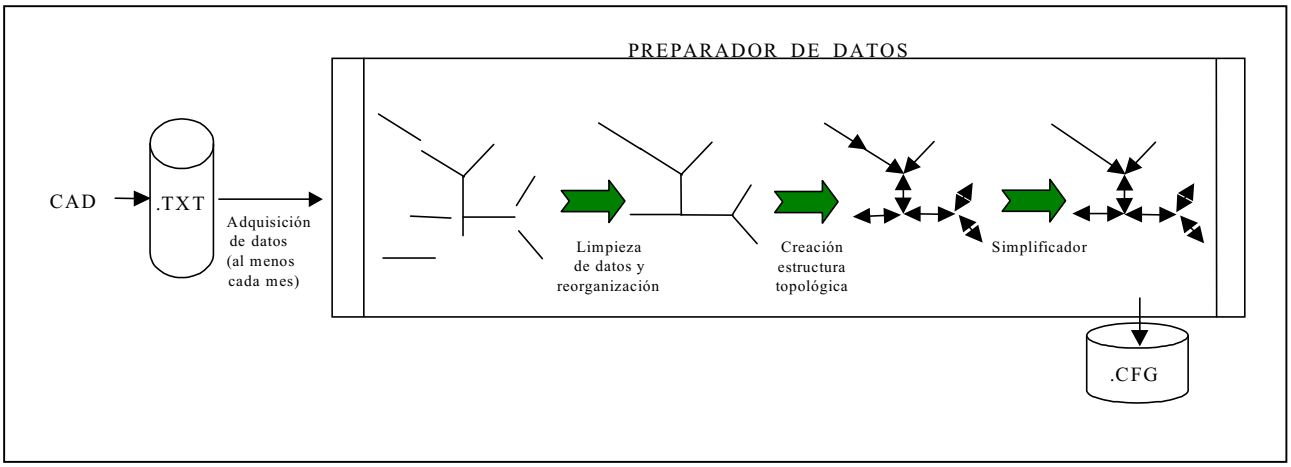

Figura 2. Pasos en el proceso de pre-tratamiento para la preparación de datos para el simulador.

2.1. Adquisición de datos.

En el caso que nos ocupa, la información de partida la constituyen los tramos de las vías que forman una red que conecta las distintas zonas de producción de material con zonas donde el material es evacuado. Lo normal es que los datos se encuentren almacenados en algún tipo de sistema CAD (en nuestro caso concreto, el sistema Microstation). En este sistema CAD, cada vía está representada por una recta (las coordenadas de dos puntos). Paralelamente, se asigna a cada tramo toda aquella información que se considere necesaria para la fase de optimización.

Lo usual es que el sistema CAD sea capaz de organizar la información por capas, de forma que podemos filtrar aquello que no se desee. Posteriormente, deberá exportarse la información a un fichero de texto, que servirá de interface con el resto del sistema. La información necesaria para su posterior uso son las coordenadas de cada vía de la red. El fichero de texto, por lo tanto, tendrá una fila para cada recta de la red conteniendo cuatro coordenadas.

A partir del fichero de texto generado desde el sistema CAD se cargan los tramos de vías. Se lee secuencialmente cada línea del fichero de texto y se crea un tramo con las coordenadas leídas en cada una de esas líneas, almacenándolo en una estructura de datos adecuada. Paralelamente se puede asignar a cada tramo toda aquella información que se considere necesaria.

2.2. Limpieza topológica y reorganización.

Cuando se lee este tipo de información desde un sistema CAD, es usual la existencia de datos con problemas de tipo topológico. Éstos son comunes en otras disciplinas, como en los sistemas de información geográfica, donde este tipo de problemas provocan que por ejemplo los polígonos no se cierren, dando lugar a áreas computacionalmente abiertas, anulando así la posibilidad de medir superficies, etc.

Los problemas topológicos más comunes que se advierten (y para los cuales hemos definido algoritmos de decisión) son los representados en la figura 3, siendo los tres primeros (*node mismatch*, *overShot* y *underShot*) los que más frecuentemente han aparecido.

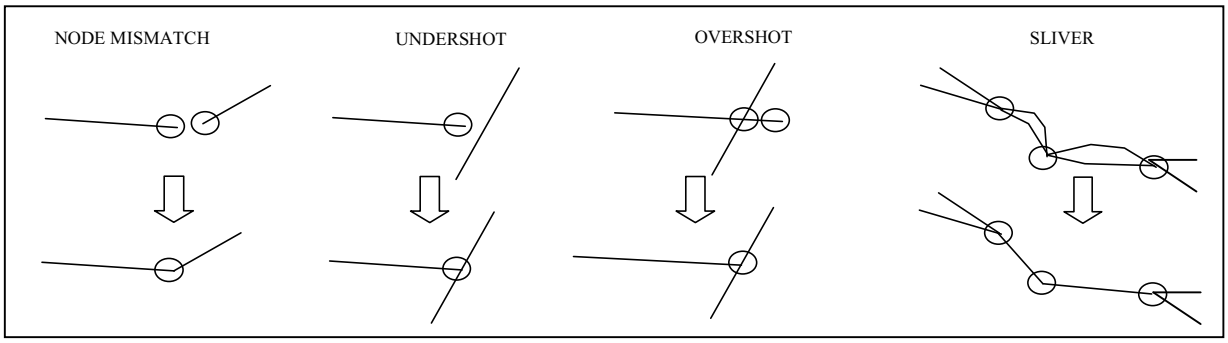

Figura 3. Problemas topológicos en el preprocesado debidos a inexactitud de los datos importados del sistema CAD.

2.2.1. Filtrado de puntos cercanos (*node mismatch*)

Este error se produce cuando tramos que deberían compartir un extremo no lo hacen, lo cual provoca que posteriormente ambos tramos no se reconozcan como adyacentes. Con el fin de identificarlos, para cada tramo se deben chequear cuatro propiedades: Comprobar si su punto inicial está a menos distancia que una determinada tolerancia ε_c del punto inicial y final de cualquier otro tramo, y lo mismo con su punto final. Cuando se detectan dos extremos con una distancia entre ellos menor que la tolerancia, es necesario mover uno de ellos sobre el otro. Pero al realizar esta operación es muy importante comprobar si el extremo que se está desplazando es común a otros tramos (y en este caso reflejar el cambio en estos tramos también), pues de lo contrario estaríamos creando otro problema de puntos cercanos.

Cada uno de estos cuatro apartados supone realizar comparaciones con cada uno de los otros tramos. Por tanto, teniendo en cuenta que este procedimiento se debe realizar para cada tramo, resulta que el filtrado de puntos cercanos es un algoritmo de orden $O(n^2)$.

2.2.2. Filtrado de *overShots* y *underShots.*

En el proceso de limpieza de *overShots* y *underShots* es necesario en algunos momentos romper tramos en dos, lo que equivale a destruir una entidad tramo y crear dos nuevas en su lugar. Para solucionar los *overShots* y *underShots* hemos definido el siguiente algoritmo:

- Para cada tramo (llamémoslo "tramo test", en la figura 4 el comprendido entre los puntos A y B, siendo A el punto inicial y B el final), se realizan los siguientes pasos:

  - Se considera la recta definida por el vector AB. Se calculan los puntos de corte de dicha recta con el resto de tramos, considerados como segmentos. En la figura 4 dicha recta corta a 4 segmentos (a, b, c y d).
  - De entre todos los tramos intersecados se toman dos: por un lado, aquél que, en el sentido del vector AB, tenga la mínima distancia al punto de final del tramo test (en el ejemplo, el tramo c), y por otro lado, aquél que, en la dirección opuesta (vector BA), que tenga la mínima distancia al punto final del tramo test (en el ejemplo de la figura el tramo b).
  - Si el tramo de corte detectado en la dirección del vector director (tramo c en el ejemplo) está a una distancia del punto inicial del tramo test menor que la longitud del tramo test

(distancia medida entre el punto inicial y el punto de corte), estamos en un caso de *overShot*.

- Si el tramo de corte en la dirección del vector director (tramo c) está a una distancia del punto inicial (punto A) mayor que la longitud del tramo pero está a una distancia del punto final (punto B) menor que cierta tolerancia ε_u, estamos en un caso de *underShot*. El procedimiento en este caso consiste en mover el punto final del tramo test hacia el punto de corte y romper el tramo cortado en dos tramos por dicho punto.
- Si el tramo de corte detectado en la dirección opuesta al vector director (tramo b) está a una distancia del punto final del tramo test (punto B) menor que la longitud del tramo test, estamos en un caso de *overShot*. El procedimiento en este caso consiste en romper ambos tramos en dos tramos usando como punto de separación el punto de corte entre ambos. En el ejemplo la solución pasa por dividir el tramo b por el punto X en dos tramos b1 y b2 y dividir el tramo AB en dos tramos AX y XB.
- Si el tramo de corte detectado en la dirección opuesta al vector director está a una distancia del punto final del tramo test mayor que la longitud del tramo test pero a una distancia del punto inicial menor que la tolerancia ε_u, estamos en un caso de *underShot*.

La parte más pesada de este procedimiento está en que para cada tramo hay que calcular puntos de corte con el resto de tramos, lo cual supone una complejidad $O(n^2)$.

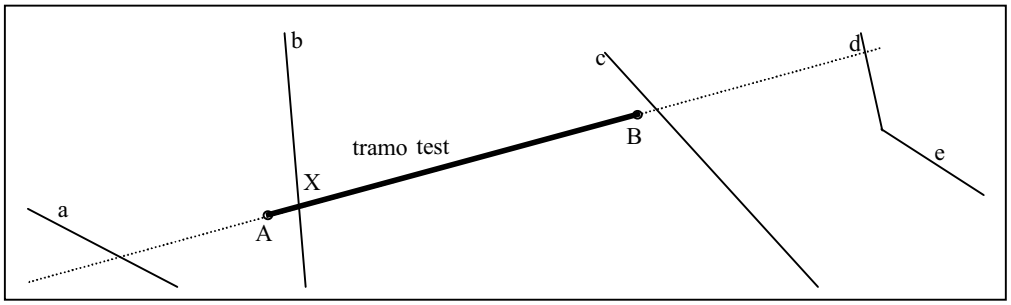

Figura 4: Ejemplo de aplicación del algoritmo desarrollado para el filtrado de o*verShots* y u*nderShots*

En el proceso de limpieza de *overShots* y *underShots* en algunos momentos es necesario romper tramos en dos, lo cual supone destruir una entidad tramo y crear dos nuevas en su lugar. Esto puede provocar inconsistencias en la estructura de datos usada (creación de posiciones vacías). Asimismo, al destruir tramos, los índices de los tramos ya no son correlativos. Es conveniente entonces realizar una reorganización de la estructura usada, para mejor funcionamiento de los algoritmos.

2.3. Creación de la estructura topológica.

En este punto del proceso, disponemos de un conjunto de tramos "limpios", lo cual no significa que todo tramo esté unido a otros dos o con un extremo libre (eso sí, nunca puede tener los dos libres, pues entonces sería inconexo y se sacaría un mensaje de alarma para su corrección manual), sino que sólo se garantiza que si dos tramos son adyacentes, deben compartir un punto extremo.

Hasta este momento se ha estado trabajando directamente con las rectas que se importaron desde el CAD. Cada recta viene definida por un punto inicial y un punto final no importando en principio cual considerar inicial y cual final. Sin embargo, a la hora de proporcionar la estructura topológica sí es importante el sentido de las vías (las relaciones "antecesor de", "sucesor de", e "inverso de").

El proceso de creación de inversos consiste en crear para cada tramo existente (siempre que la circulación sea en ambos sentidos) su tramo inverso, que será un nuevo tramo (por lo tanto con un identificador nuevo para él) cuya coordenada inicial será la coordenada final del tramo y cuya coordenada final será la inicial del tramo. Es conveniente guardar en algún campo de la estructura de datos de un tramo el identificador del tramo inverso de forma que sea rápido y sencillo el acceder a él, ya que será útil para el optimizador.

Las relaciones "sucesor de" y "antecesor de" son recíprocas y por lo tanto no requieren cálculos separados. El proceso de cálculo es sencillo: para cada tramo se buscan sus tramos sucesores, que serán aquellos cuyo punto inicial coincide con el punto final del tramo excepto el tramo inverso (con una tolerancia máxima ε_i). Cuando se encuentre un tramo en este caso, se almacena como tramo sucesor y al mismo tiempo se indica aquel tramo es antecesor del encontrado. La complejidad de este proceso es como en casos anteriores $O(n^2)$.

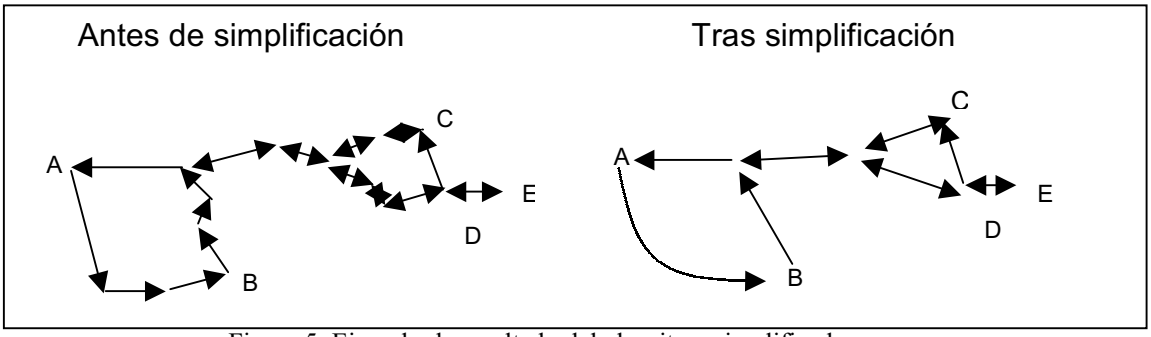

Figura 5: Ejemplo de resultado del algoritmo simplificador

2.4. Simplificación de la información.

Este último paso, la simplificación de la red, permite mantener la información necesaria ahorrando, en almacenamiento y tiempo posterior de cálculo. Mantener una red con varios cientos de tramos dentro de un algoritmo de optimización en el cual se debían calcular varias simulaciones de toda la red, dentro de las cuales a su vez se debían calcular rutas varias decenas de veces, supone un gran costo computacional.

Sin embargo, para estos cálculos en principio no es necesario conocer todo el nivel de detalle que proporciona el CAD. La información interesante en una red decisional de transporte es el cómo se conectan los tramos para configurar rutas. Los puntos interesantes son aquellos que plantean una posible decisión de qué camino tomar, es decir, las bifurcaciones. La idea es que no hay por qué mantener una cadena de tramos en los que no hay bifurcación ya que no ofrecen ninguna información de cara a la ruta a seguir. Dicha cadena de tramos puede ser sustituida por un solo tramo que mantenga la información necesaria con el consiguiente ahorro (figura 5). Los pasos del algoritmo simplificador son:

- Mientras la lista de tramos contenga tramos simplificables (un tramo es simplificable si tiene un único sucesor y tanto él como el sucesor no contienen zonas que pueda ser origen o destino de una ruta), seleccionar uno y hacer:
  - Crear un nuevo tramo y asignarle el mismo identificador que el tramo, y como longitud la suma de las longitudes de ambos tramos (y no la euclídea entre extremos).
  - Asignarle como sucesores los del segundo tramo y actualizar la información en esos sucesores.
  - Asignarle como antecesores los del primero y actualizar la información en esos antecesores.

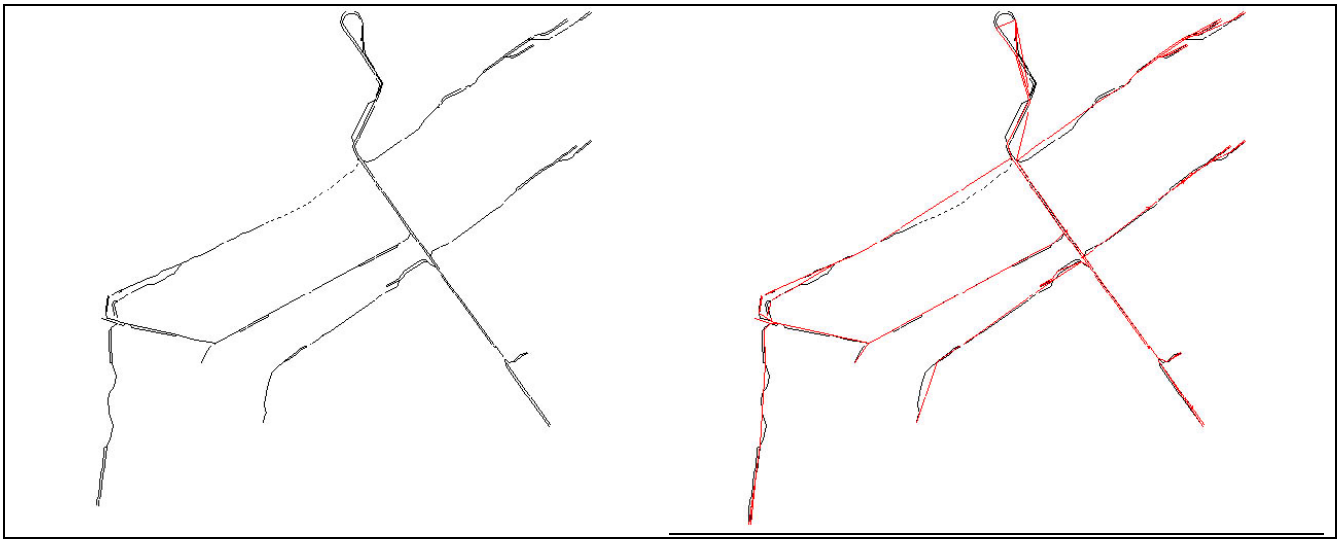

Figura 6: Configuración típica (izquierda) de tramos de la mina Samuño vs. tramos simplificados

Al ser éste un proceso destructivo (la lista original se pierde durante el proceso), es conveniente realizar una reorganización tras la simplificación, ya que pueden quedar posiciones de memoria vacías.

2.5. Actualización de la información.

Como se ha indicado, con cierta periodicidad es necesaria la actualización de la información debido a cambios físicos en la red de transporte interno. Como mínimo una vez al mes estos cambios se reflejan en el sistema CAD. Para poder recoger estos cambios sin destruir la información que ya podamos tener almacenada en nuestra red, cada vez que se actualiza, el algoritmo diseñado compara cada tramo de la nueva red con los de la antigua, decidiendo la estructura final.

3. Implantación en entornos reales.

La metodología diseñada para la carga automática de datos para el funcionamiento del optimizador ha sido probada en varias minas simulando varias semanas reales con el fin de comprobar su funcionamiento. En dichas pruebas se usaron las siguientes tolerancias:

1. Para el filtrado de puntos cercanos se usó una tolerancia ε_c=0.5 metros.
2. Para la detección de o*verShots* y u*nderShots* se utilizó una tolerancia ε_u=0.1 metros.
3. Para la de inversos, sucesores y antecesores se usó una tolerancia ε_i=0.01 metros.

Se observa que las tolerancias usadas se eligieron de modo que van disminuyendo en cada paso, debido a que los datos son cada vez más precisos y correctos según el proceso se ejecuta. En la tabla 1 se observa la notable reducción en el tamaño del problema tras aplicar el algoritmo simplificador en tres minas distintas.

Antes de simplificación	Después de simplificación
716	232
924	346
986	350

Tabla 1: Reducción del número de tramos al aplicar el algoritmo

La figura 6 muestra, para el primer ejemplo de la tabla 1 (mina Samuño), la comparación gráfica dada por el simulador entre la configuración inicial y la final simplificada.

4. Conclusiones

Cuando se manejan sistemas con grandes volúmenes de información (como ocurre en proyectos con información geográfica) es imprescindible diseñar algoritmos que preparen los datos de forma automática, con el doble objetivo de que los datos de entrada al sistema sean fiables, y sin información superflua (para que los cálculos posteriores sean lo más rápidos posible). En este trabajo se ha presentado un caso de aplicación de preprocesamieto dentro de un sistema de optimización de políticas de transporte interno en minería, mediante simulación. Se han presentado los pasos que se han visto necesarios para lograr ambos objetivos, y se validaron en un entorno real de varias minas.

Referencias

[1] Law, A. M. y W. D. Kelton (2000). *Simulation modeling and analysis*. McGraw, Singapore.

[2] Sturgul, J.R. (2000). *Mine Design: Examples using Simulation*, Society for Mining, Metallurgy, and Exploration, Inc. Littleton, Colorado.

[3] Arsham, H., A. Feuerverger y D.L. McLeish (1989). “Sensitivity analysis and the ‘what if’ problem in simulation analysis”. *Mathematical and Computer Modelling*, Vol. 12, No. 2, 193-219.

[4] Schiess, C. (1993). “Simulation: How to set goals and how to get started.”. *Industrial Engineering*, Vol. 25, No. 5, 26-27.