


```
//left offset of cutdepth at ct
settablenum(offset_table,11,2,cutdepth*sin(degreestoradians(CT_degreeR)));
//right offset of cutdepth at ct
```

6.2.6 The Shuttle Car (SC)

The SC was modelled as a *task executer* object in Flexsim®. *Task executers* can travel, load *flowitems*, unload *flowitems* and perform many other simulation tasks. The *task executer* used in this model can transport coal *flowitems* from the CM to the BE. The easiest way to use a *task executer* in Flexsim® is to connect the *task executer* (the SC) with the CM using the centre connection method and check the “**Use Transport**” option (Figure 6-24). The default method cannot manage the complexity of the SC travelling routes, which is handled using customised task sequences in the codes area of **Request Transport From** (when a SC is loaded and requested) and the SC **OnUnload** trigger (when the SC is unloaded).

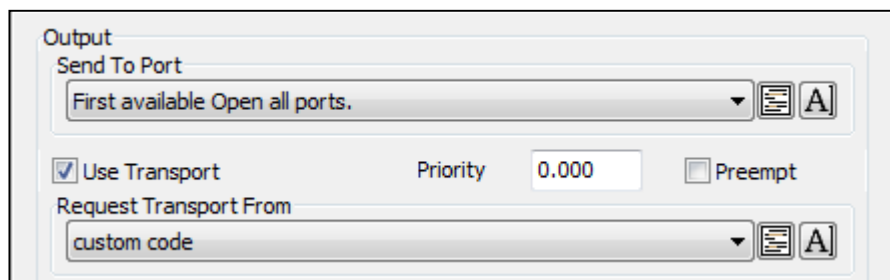


Figure 6-24 The Flow page of CM properties window

6.2.6.1 Shuttle Car travelling routes

The SCs are largely constrained by the roadway layout and mine safety issues. The interactions between SCs are unlike cars on a surface road because underground space is very narrow. Moreover, a SC is powered by electricity through a cable linked between itself and a hook anchored to the side of the roadway. SCs cannot cut across each other or pass in the heading; they must wait for others at a specific location, usually near the intersection of a cut-through.

The routes have a huge impact on how the SCs interact and the distance from the CM to the BE, which adds other practical complexities to the model. All possible routes can be seen as two parts: from the CM to the point before going into the intersection of the cut-through (CM part) and the left part to the BE (BE part) (Figure 6-25).

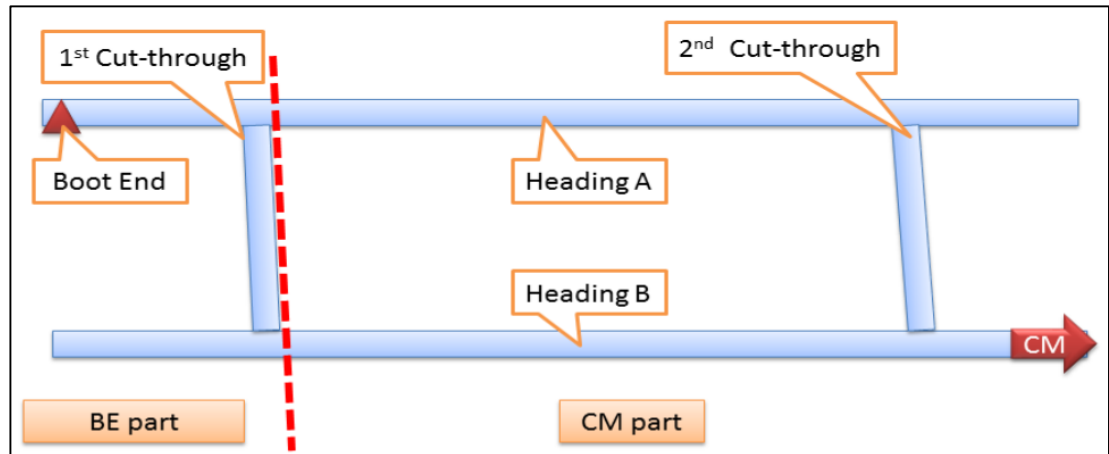


Figure 6-25 The two parts of SC travelling routes

For the CM part, the SC has different routes when the CM is in the heading and when the CM is in the cut-through. There are three instances depending on the location of the CM: heading, right hand side cut-through and left hand side cut-through (Figure 6-26). When the CM is in the heading, the SC travels directly to the CM's location (for example in Figure 6-27, the red lines labelled 1 and 3). When the CM is in the cut-through, the SC travels a full pillar length before travelling to the CM's location.

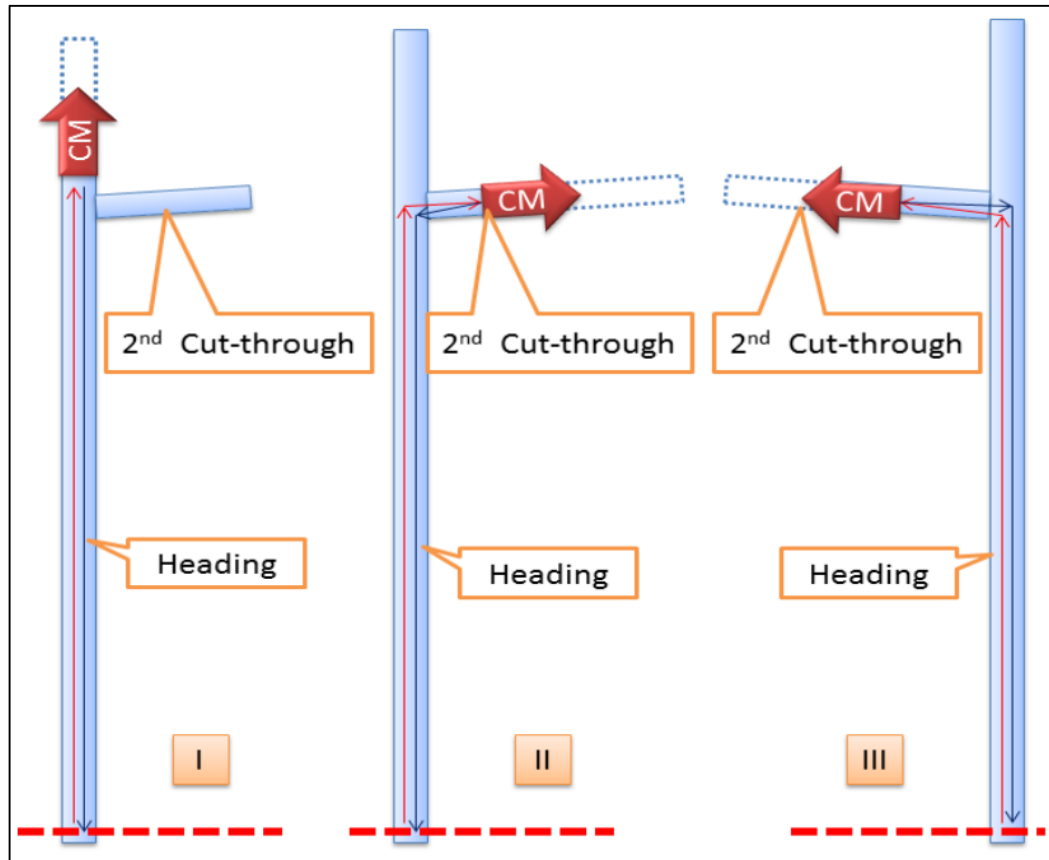


Figure 6-26 The three cases of SC travelling routes of the CM part

Figure 6-27 to Figure 6-31 illustrate the SC tramming routes when two SCs were used and a CM is in the heading. SC1 follows the red routes; SC2 follows the blue routes. The red routes are when only one SC is used for one CM. The routes are divided into several steps and between steps, the SC either stops to wait for the other SC or sends a message to start the other SC which is waiting. For modelling purposes, the routes can be classified as follows when the CM is in the heading:

- i. one CM:
 - a) the CM is in the same heading as the BE (Figure 6-27):
 - SC2 waiting area is in the right hand side cut-through (Left of Figure 6-27);
 - SC2 waiting area is in the left hand side cut-through (Right of Figure 6-27).

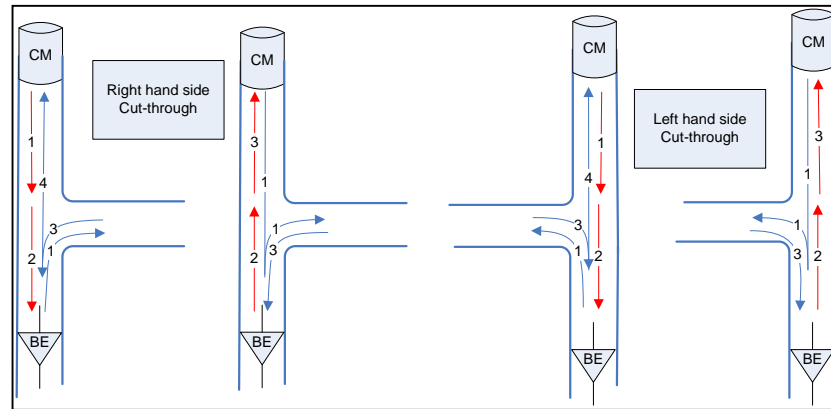


Figure 6-27 SC travelling routes (CM is in the BE heading)

As shown in Figure 6-27, the routes are divided into four steps. The same number means the two SCs could begin to travel at the same time. SC1 (the red routes) must check whether the BE is occupied before going into the BE area and send a message to SC2 waiting aside immediately when coming into the BE area. After SC1 finishes dumping, it waits at the end of the BE area until SC2 reaches its waiting area from the CM. When SC1 leaves the BE area, it also signals SC2 which is waiting to go into the BE area. After dumping, SC2 must not go to the CM, but to its waiting area until it receives a message from SC1. When SC2 is loaded, it goes to the waiting area again and sends a message to SC1 if SC1 is waiting at the end of the BE area.

- b) the CM is not in the same heading as the BE: The routes are divided into three steps. SC2 (blue routes) must wait at the waiting area before going into or when leaving the BE area. There are two cases as follows:

- CM is in the left hand side heading from the BE heading (Figure 6-28);

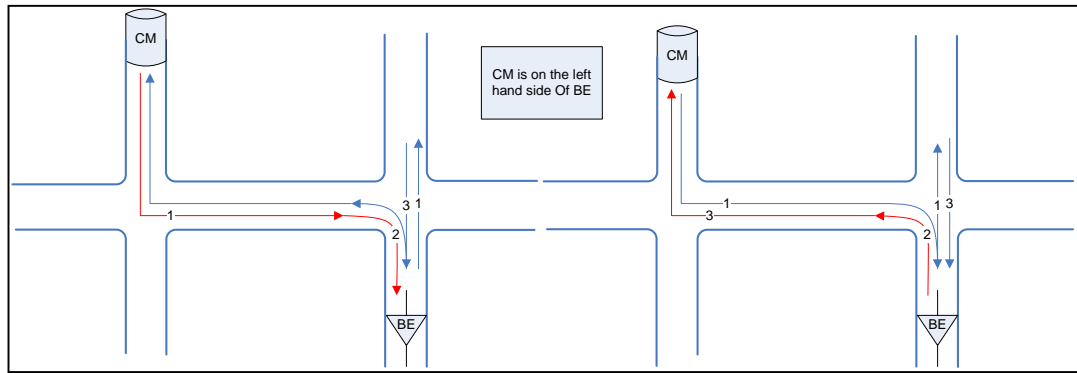


Figure 6-28 SC travelling routes (CM is in left hand side heading)

- CM is in the right hand side heading from the BE heading (Figure 6-29).

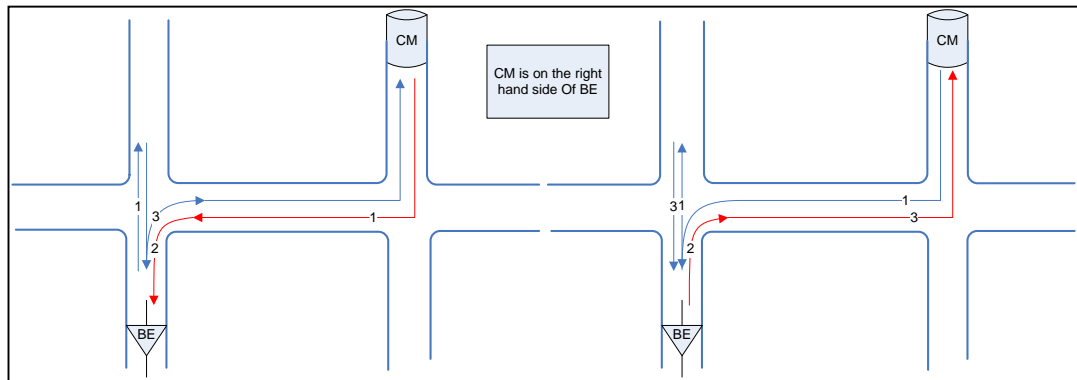


Figure 6-29 SC travelling routes (CM is in the right hand side heading)

- ii. two CMs and two SCs: Each SC must check whether the BE is occupied before going into the BE area. If the BE area is occupied by the other SC, it should wait at the end of the intersection until it receives a message indicating the other SC has left the BE area. There are two situations:
 - a) the second CM is in the left hand side heading from the BE heading (Figure 6-30);

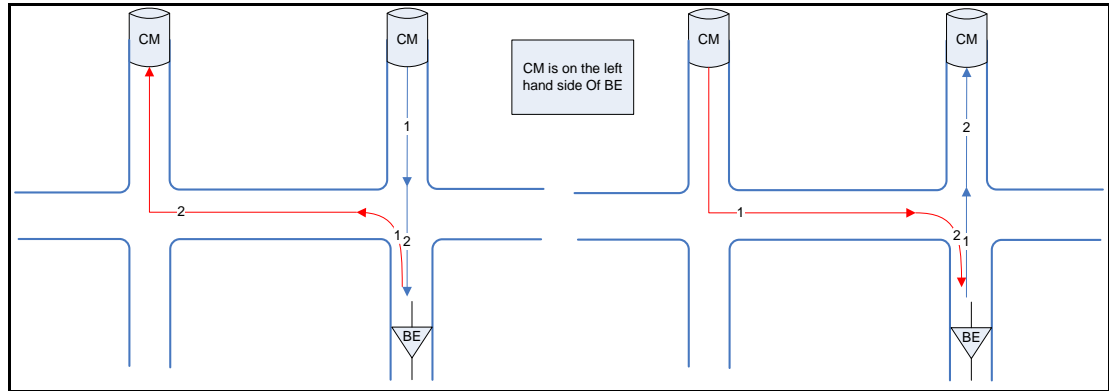


Figure 6-30 SC travelling routes – the 2nd CM is in the left hand side heading

b) the second CM is at the right hand side heading from the BE heading (Figure 6-31).

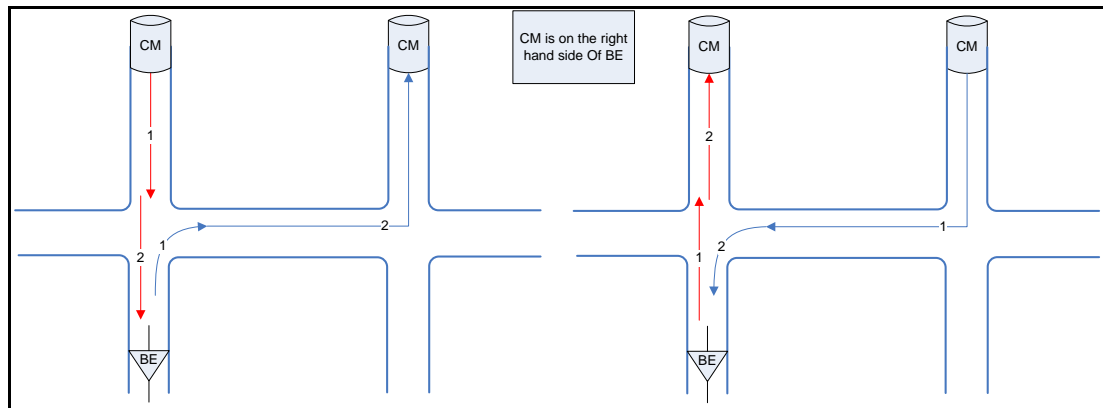


Figure 6-31 SC travelling routes – the 2nd CM is in the right hand side heading

6.2.6.2 The scripts for SC routes

To simplify writing the scripts, five *Flexscript* nodes were created and stored in the variable treenode of the CM object ranked from 16th to 20th (Figure 6-32). To create a script node, a normal node was created first by using the spacebar when highlighting the 15th node and then right clicking on the newly created node and selecting **Build | Toggle Node as FlexScript**. The codes can be written inside the *Flexscript* node in the same way as the triggers.

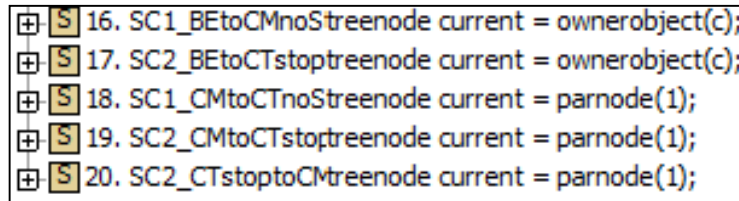


Figure 6-32 The *Flexscript* nodes of SC travelling routes under CM variable tree

The functions of the respective nodes are:

- the 16th node: dispatch the first SC linked to the CM travelling from the BE to the CM;
- the 17th node: dispatch the second SC linked to the CM travelling from the BE to the waiting point near the first cut-through;
- the 18th node: dispatch the first SC linked to the CM travelling from the CM to the waiting point near the first cut-through;
- the 19th node: dispatch the second SC linked to the CM travelling from the CM to the waiting point near the first cut-through;
- the 20th node: dispatch the second SC linked to the CM travelling from the waiting point near the first cut-through to the CM.

As may be noticed, there is no node to dispatch the SC travel to the BE and unload the coal *flowitem* to the BE. This must be done in the BE OnMessage trigger.

The five nodes can be divided into two categories:

- i. those that dispatch the SC from the CM (the 18th and the 19th): Before the SC leaves the CM, the SC must be loaded with the coal *flowitem* processed by the CM. So the header of the two nodes are three involved parameters as follows:

```

treenode current = parnode(1); // should be the CM
treenode item = parnode(2); // should be the coal flowitem
int port = parval(3); // the port index of the CM linked to the SC

```

To execute the nodes, use the command `nodefunction(fstravelsc, tonum(current), tonum(item), SCindex)`, where *fstravelsc* refers to the 18th

node or the 19th node; `tonum(current)` is the CM object; `tonum(item)` is the coal *flowitem* and *SCindex* is the port index of the CM linked to the SC;

- ii. those that dispatch the SC travel towards the CM (the 16th, the 17th and the 20th):

There is no load and unload task. The SC only travels to its destination and sends a message to the other SC. The header of the nodes is as follows:

```
treenode current = ownerobject(c); // should be the CM
treenode transporter= i; // should be the SC
```

To execute the nodes, use the command `executefsnode(fstravelsc, c, i)`, where `fstravelsc` refers the FlexScrip node; the owner object of the node is the CM. Actually, the two commands `executefsnode` and `nodefunction` have the same function to execute a *Flexscript* node, but `nodefunction` has more customised parameters. For more information about the two commands, please refer to the Flexsim (2013).

The five nodes use the same idea to handle different cases of SC travelling routes: calculate the landmark points, then create a series of task sequences and dispatch it.

Four task types were used:

- `TASKTYPE_FRLOAD`: this task loads a *flowitem* from a fixed resource (CM) to a task executer (SC);
- `TASKTYPE_TRAVELTOLOC`: this task makes a task executer (SC) travel to a specified location;
- `TASKTYPE_SETNODENUM`: this task sets a node value that is used to change the SC direction when the SC enters a cut-through from a heading or enters a heading from a cut-through;
- `TASKTYPE_SENDMESSAGE`: This task sends a message to an object.

6.2.6.3 The triggers involved in the SC routes

To simulate the travelling routes, five triggers were used in the Flexsim model:

- i. the CM **Pick Operator** codes area: stops the CM from cutting if the SC is not ready;
- ii. the SC **OnReset** trigger: sets the initial location of the SC when the model is reset;

```
if( cpcpno(current,1)>1)
// the second SC
{
    //set its initial location at the BE area,
    //and dispatch the SC to its waiting area
}
else // the first SC of CM, set location to the CM
{
    // set its initial location to be the rear of the CM
}
```

- iii. the CM **Request Transporter From** codes area: dispatches the SC travelling from the CM or waiting area to the BE and unloads the coal *flowitem* when the cutting and loading process is completed (the actual loading time of the SC object was set to be 0);

```
if(nrcp(current)==1)//only one SC
{
    fstravelsc=node(">18",current); // SC1 from the CM to the BE
}

else //two SCs
{
    if (SCindex==1 || nrcp(current)==1) // the first SC
    {
        fstravelsc=node(">18",current);
        // SC1 from the CM to the BE
        setlabelnum(current,17,2);
        // SC1 leaves, set current SC to be SC2
    }
    Else // the second SC
    {
        fstravelsc=node(">19",current);
        // SC2 from the CM to the waiting area
        setlabelnum(current,17,1);
        // SC2 leaves, set current SC to be SC1
    }
}

nodefunction(fstravelsc,tonum(current),tonum(item),SCindex);
// executes the SC travelling node
setlabelnum(current,18,0);
```

```
// SC leaves, set SC not ready to be loaded
```

- iv. the SC **OnUnload** trigger: dispatches the SC travelling from the BE to the CM or waiting area when it is unloaded:

```
if (SCindex==1) // the first SC
{
    if(!getlabelnum(CM,5))//CM is not occupied
    {
        setlabelnum(station,3,0);//no SC waiting at BE area
        fstravelsc=node(">16",CM); //Dispatch SC1 from BE to CM
        executefsnode(fstravelsc,fstravelsc,current);
    }
    else //CM is already occupied
        setlabelnum(station,3,tonum(current));
        //record itself to restart (set the BE being occupied).
        //The SC waits at the BE
}

else if (SCindex==2) // the second SC
{
    fstravelsc=node(">17",CM);
    // Dispatch SC2 from BE to waiting area
    executefsnode(fstravelsc,fstravelsc,current);
}
```

- v. the BE **OnMessage** trigger: dispatches a SC travelling to its destination in case it is waiting somewhere.

```
case 1: //sender:SC1_CMtoCTnoStop && SC2_BEtoCTstop
    //&& SC1_CMtoCTnoStop (&& SC1_BEtoCMnoStop if 2 CMs),
{
    //SC1 from waiting point to BE to unload,
    //please refer to the model
    break;
}
case 2: //sender: SC1_BEtoCMnoStop,to resume shuttle car 2 of CM1,
{
    //SC2 from waiting point to BE to unload
    //please refer to the model
    break;
}
case 3://sender: SC2_CMtoCTstop
{
    treenode transporter=tonode(msgparam(2));//sender object
    setlabelnum(centerobject(transporter,1),5,0);
    //CM not been occupied
    int SC_numID=getlabelnum(current,3);
    if (SC_numID)
    {
        treenode fstravelsc=node(">16",inobject(current,1));
        executefsnode(fstravelsc,fstravelsc,tonode(SC_numID));
    }
    break;
}
case 4://sender: case 1
{
    treenode transporter=centerobject(inobject(current,1),2);
    //2nd SC of CM1
```

```

        setlabelnum(centerobject(transporter,1),5,1);
        //CM not been ocupied
        treenode fstravelsc=node(">20",inobject(current,1));
        nodefunction(fstravelsc,centerobject(transporter,1),
                    first(transporter),2);
        break;
    }

    default: break;

```

6.2.6.4 *Some important strategies for the SC routes*

The following approaches were used to monitor the SC routes:

- i. to determine whether a working heading is in the heading to the left or right hand side of the BE heading: subtract the heading number from the number of the BE heading. If the difference is 0, the CM is in the BE heading; if less than 0, it is in left hand side heading, while more than 0 means the CM is in the right hand side heading;
- ii. to determine whether the CM is working in a cut-through or a heading: calculate the difference in direction for the CM and the roadway (as a whole unit). If the difference is 0, the CM is in the heading; otherwise, the CM is working in the cut-through. When the CM is in a cut-through, find the value in the heading column of the current task in the task sequence table. If the decimal part of the value is 0.1, the cut-through is to the right hand side of the heading, while 0.2 means the left hand side cut-through;
- iii. to determine whether a CM is served by more than one SC, use the command `nrcp` to find the total centre connections of the CM object which is the total number of SCs used for the CM.

6.2.6.5 Setting the SC empty and full loaded frame

Two 3D files were imported into Flexsim® through the menu **Tools | Media Files**, one was called *ShuttleCar.skp* for the empty SC, the other was called *ShuttleCarFRAME1.skp* for the loaded SC (Figure 6-33). After the SC was loaded (SC **OnLoad** trigger), the command `setframe(current,1)` was used to set the 3D view of the SC in the 3D environment as the loaded one; and after the SC was unloaded (SC **OnUnload** trigger), the command `setframe(current,0)` was used, which resulted in the SC view being the default 3D view, which is empty. When the model was reset, the SC was set to be empty by the SC **OnReset** trigger.



Figure 6-33 SC loaded and empty frames

6.2.7 The use of a Continuous Haulage System (CHS)

To simulate the use of a CHS, one easy way is to remove the SCs and uncheck the options **Use Operator(s) for Process** and **Use Transport**, then the coal *flowitem* goes directly to the BE without any delay. This does not really happen when a CHS is used in a coal mine, but the impact of the CHS capacity can be set up by changing the CM

cutting process time per web. To make the model easy to use, the following scripts were written in the CM **OnReset** trigger to set whether to use an SC or not, depending on a label value.

```
int shuttlenum=getlabelnum(current,14); // label value of SC number
setnodenum(node("/usetransport",variables(current)),shuttlenum);
// Use Transport, if shuttlenum=0, uncheck the option, otherwise, check
it.
setnodenum(node("/useprocessoperators",variables(current)),shuttlenum);
// Use Operator(s) for Process, if shuttlenum=0, uncheck the option,
// otherwise, check it.
for (int i=1; i<=nrcp(current);i++)
//hide SCs if not use them (shuttlenum = 0), otherwise show them
{
    switch_hideshape(centerobject(current,i),!shuttlenum);
    switch_hideconnectors(centerobject(current,i),!shuttlenum);
}
```

6.2.8 *The roadway*

The roadway, which is actually the outcome of a roadway development system is a very special object. It extends as mining advances but it is not one-directional like a motorway, its shape depends on the design of the roadway layout. A good aspect of Australian roadway development is the layouts usually entail mostly repeated blocks. In the modelling work, nine variables of roadway development geometry were considered, with all the data being stored in the labels of the roadway object. These variables are:

- i. the total number of headings;
- ii. the location heading of BE;
- iii. the angle of cut-through;
- iv. the width of the pillar;
- v. the length of the pillar;
- vi. the width of the roadway section;
- vii. the height of the roadway section;
- viii. the overdrive length; and
- ix. the distance from the BE to the cut-through.

Flexsim[®] does not have an object such as a roadway shape which changes its shape as the CM advances. Such an object was designed using a *Visual tool* object in Flexsim[®] and a 3D view of the object was drawn by customised codes located in the roadway object's **OnDraw** trigger (**Custom Draw Code**). Two key draw methods were used:

- i. `drawcube`: this command draws a cube with its origin positioned at its bottom left corner. Its length, width and height respectively correspond to the length, width and height of a section of roadway. This command is easier to use than the other method and was used to draw the headings. There were two different cases involved in drawing the headings:
 - a section of heading with a fixed length, such as finished headings and finished tasks in the task sequence table (a task means a section of roadway);
 - a section of roadway with a changing length in which a CM is working. The length was the length accumulated from when the task started.
- ii. OpenGL method: `glBegin(GL_QUAD_STRIP)` was used to mark the beginning of a set of `glVertex3d()` commands. The `glVertex3d()` command was used to define the vertices of the specified shape. Using the `GL_QUAD_STRIP` primitive method, the first four vertices created a quad and every two vertices after that created a quad with the previous two. The set of `glVertex3d()` commands must be followed by the `glEnd()` command to mark the completion of the primitive definition. The command `glVertex3d()` has three variables to define the position of the vertex in the 3D environment.

This method was used to draw cut-throughs because they may not be at right angles to the headings, making a non-rectangular shape and the `drawcube` command cannot handle this situation.

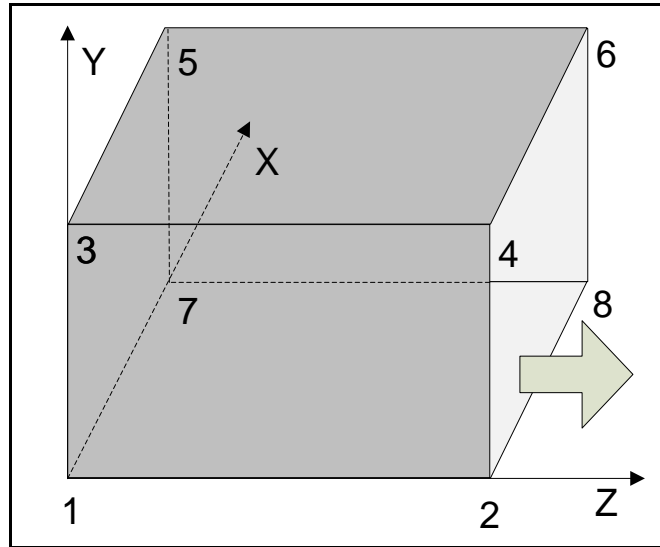


Figure 6-34 Using OpenGL method to draw the 3D view of the roadway

As Figure 6-34 shows, the cube was defined by 8 dots and each dot was defined by three values of location. This gives enough flexibility to draw a roadway of any standard shape. An example of a section of codes is provided below:

```
glBegin(GL_QUAD_STRIP);
glColor4d(cR, cG, cB, Opacity2); // set color and opacity
glVertex3d(ctloc_offsetleft1+i_offsetleft2, 0, z_offset2); // dot 1
glVertex3d(init_ct_loc+i_offsetleft2, 0, z_offset1); // dot 2
glVertex3d(ctloc_offsetleft1+i_offsetleft2, way_height, z_offset2);
// dot 3
glVertex3d(init_ct_loc+i_offsetleft2, way_height, z_offset1); // dot 4
glVertex3d(ctloc_offsetleft1+way_width_newR+i_offsetleft2, way_height,
z_offset2); // dot 5
glVertex3d(init_ct_loc+way_width_newR+i_offsetleft2, way_height,
z_offset1); // dot 6
glVertex3d(ctloc_offsetleft1+way_width_newR+i_offsetleft2, 0,
z_offset2); // dot 7
glVertex3d(init_ct_loc+way_width_newR+i_offsetleft2, 0, z_offset1);
// dot 8
glVertex3d(ctloc_offsetleft1+i_offsetleft2, 0, z_offset2); // dot 1
glVertex3d(init_ct_loc+i_offsetleft2, 0, z_offset1); // dot 2
glEnd();
```

In this example the dots 1, 3, 5, 7 are fixed by the location from where the roadway starts and dots 2, 4, 6, 8 are the fixed four dots, 1, 3, 5, 7 plus the length of the section of roadway, which is either a fixed length of a finished part or the accumulated length of the current task. As in the longwall model, to draw a 3D view, `drawtomodelscale(current)` must be used before the draw codes start.

The following variables make the 3D roadway's colour and opacity changeable.

```
double cR=getnodenum(rank(color(current),1)); // Red value from 0 to 1
double cG=getnodenum(rank(color(current),2)); // Green value from 0 to 1
double cB=getnodenum(rank(color(current),3)); // Blue value from 0 to 1
double colorR=cR*255; // convert the value to be 0 to 255
double colorG=cG*255; // convert the value to be 0 to 255
double colorB=cB*255; // convert the value to be 0 to 255

// Opacity value stored in the roadway labels
double Opacity=getlabelnum(current,6);
double Opacity2=getlabelnum(current,7);
double Opacity3=getlabelnum(current,8);
```

The three opacity values were used to distinguish between different parts of the roadway and to make objects visible inside the roadway object. The colour values are default variables of *Visual tool* object in Flexsim®, which can be set up from the object properties window. The `drawcube` command has three parameters responding to red, green and blue values in the format of 0-255 and an opacity parameter. In the OpenGL method the following must be used, as in the previous example. The colour values of this method are from 0 to 1.

```
glColor4d(cR, cG, cB, Opacity); // set colour and opacity
```

The draw codes were divided into 3 parts, the initial part, the finished pillars and the current working part. There are three sub-parts within each part: parts to the left of the BE heading, parts to the right of the BE heading and parts to left of the BE heading. Like the design of the SC travelling routes, the key task was to calculate the landmark points (the start and end point of each section of roadway) as is discussed below:

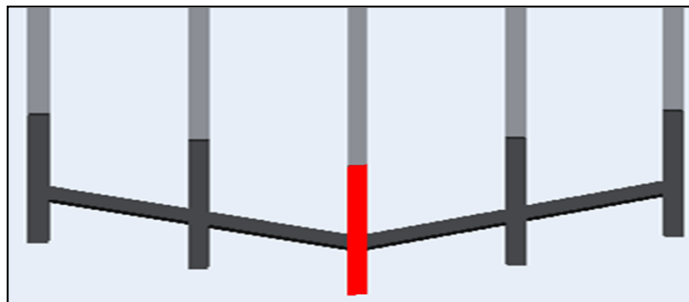


Figure 6-35 Initial roadways to draw

i. draw initial part

The initial part consists of the roadways that existed before the model began, which include the part from the BE to the 1st cut-through, the overdrive part and the 1st cut-throughs (as shown in Figure 6-35, in black and red). Originally the red part had the same colour as the parts in black in the model. The red section was added afterwards. The heading marked in red is where the BE was located.

This part was drawn by two steps:

- draw the BE heading using a `drawcube` command which is implemented as follows:

```
drawcube(0,pillar_width*(1-be_heading)+half_waywidth,0,
         xlength,way_width,way_height,0,0,0,
         colorR,colorG,colorB,Opacity2); // heading with BE
```

The starting point is $(0, \text{pillar_width} * (1 - \text{be_heading}) + \text{half_waywidth}, 0)$ relative to the origin of the roadway itself (left bottom corner). The length is the sum of the overdrive length and the distance from the BE to the cut-through, the width is the width of the roadway and the height is the height of the roadway;

- the second parts at the sides of the BE heading are symmetrical with repeated units. The repeated neighbouring units have a difference in location that is the width of a pillar and the difference in location in the heading direction is the width of a pillar multiplied by the cotangent of the cut-through angle. These units can be drawn using the **For** loop statement where each unit has part of a heading that can be drawn using the `drawcube` command and a cut-through that can be drawn using the OpenGL method.

ii. draw finished pillars

The white and red parts in Figure 6-36 form a completed unit of one pillar length of panel advance. The red and white colour were added to demonstrate only, which were originally in the same colour as the parts in grey. This part has the same features as the initial part and can be drawn using the same method.



Figure 6-36 Completed roadways plus initial roadways

The pillar lengths of panel advance are repeated units when more than one pillar length is configured, with each pillar being a repeat of the white and red parts in Figure 6-36. The number of completed pillars is the current pillar index stored in the label of the roadway, minus 1. A **For** loop statement was used to draw the repeated pillars, with another two **For** loop statements nested into it to draw the left and right parts of each pillar.

To preview the geometrical setup of the roadway a flag value was used and then stored in the label of the roadway. If the preview is true, draw all the pillars being set up with model user, but if it is false or when the model is running, only draw the completed pillar/pillars. The simplified codes are provided below:

```
if (getlabelnum(current,"preview") || cur_pillar_num > 1)
{
    int pillar_num=gettablenum(rd_geo_table,6,1);
    // the required number of pillars to develop by model user
```

```

if (cur_pillar_num > 1)
// currently developing the 2nd pillar,
// at least one completed pillar in the model
pillar_num=cur_pillar_num-1;
for (int i=0; i<=pillar_num-1; i++)
{
    // draw codes
}
}

```

iii. draw active part.

The active part is the pillar where the CM or CMs are currently working (the black part in Figure 6-37). This is the most difficult part to draw because it has an irregular shape which depends on the task sequence table (or tables if multi CMs), completed tasks and the percentage completed of current task.

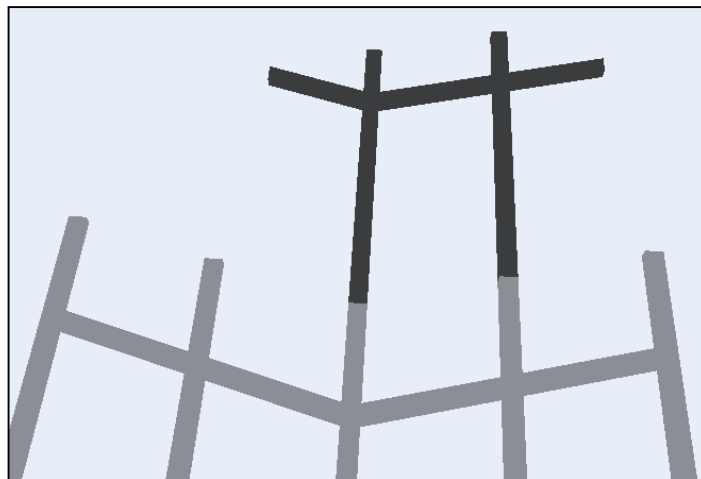


Figure 6-37 Current pillar (the black part)

- Loop through all possible task sequence tables - there may be more than one CM and each CM has its own task sequence table. All the active CMs working in these roadways must be linked to the BE object, so the number of CMs can be accessed by checking the number of BE connections. The following are the codes that were used to loop through all the tables of each CM.

```

int num_CM=content(connectionsin(bootend));
for (int i=1; i<=num_CM; i++)
// for each CM, draw roadways based on CM's task table
{
    treenode iCM=inobject(bootend,i); // the ith CM
}

```

```

treenode tasktable= label(iCM,7);
// the task sequence table of the ith CM

//draw codes
}

```

- For each CM, draw roadways of completed tasks. The completed tasks are the rows above the current task row in the task sequence table. The value of the current task index is stored in the 7th label of the CM. A **For** loop statement was used to loop through the completed tasks and the roadway for each task was determined by the *frompoint*, *topoint* and heading index of the task sequence table (Figure 6-12), in addition to the geometrical variables of the roadway, i.e. the height and width of the roadway.
- For each CM, draw the roadway of the current task like the completed task, but the roadway length of the current task is determined by the length that accumulated since the current task began.

6.2.9 Integrated flow logic of the model

Figure 6-38 shows the integrated logic of all the pieces of logic discussed above, including different miner types, cutting, loading and supporting cycles, face operation delays, multiple CMs, multiple pillars and the use of task sequence tables.

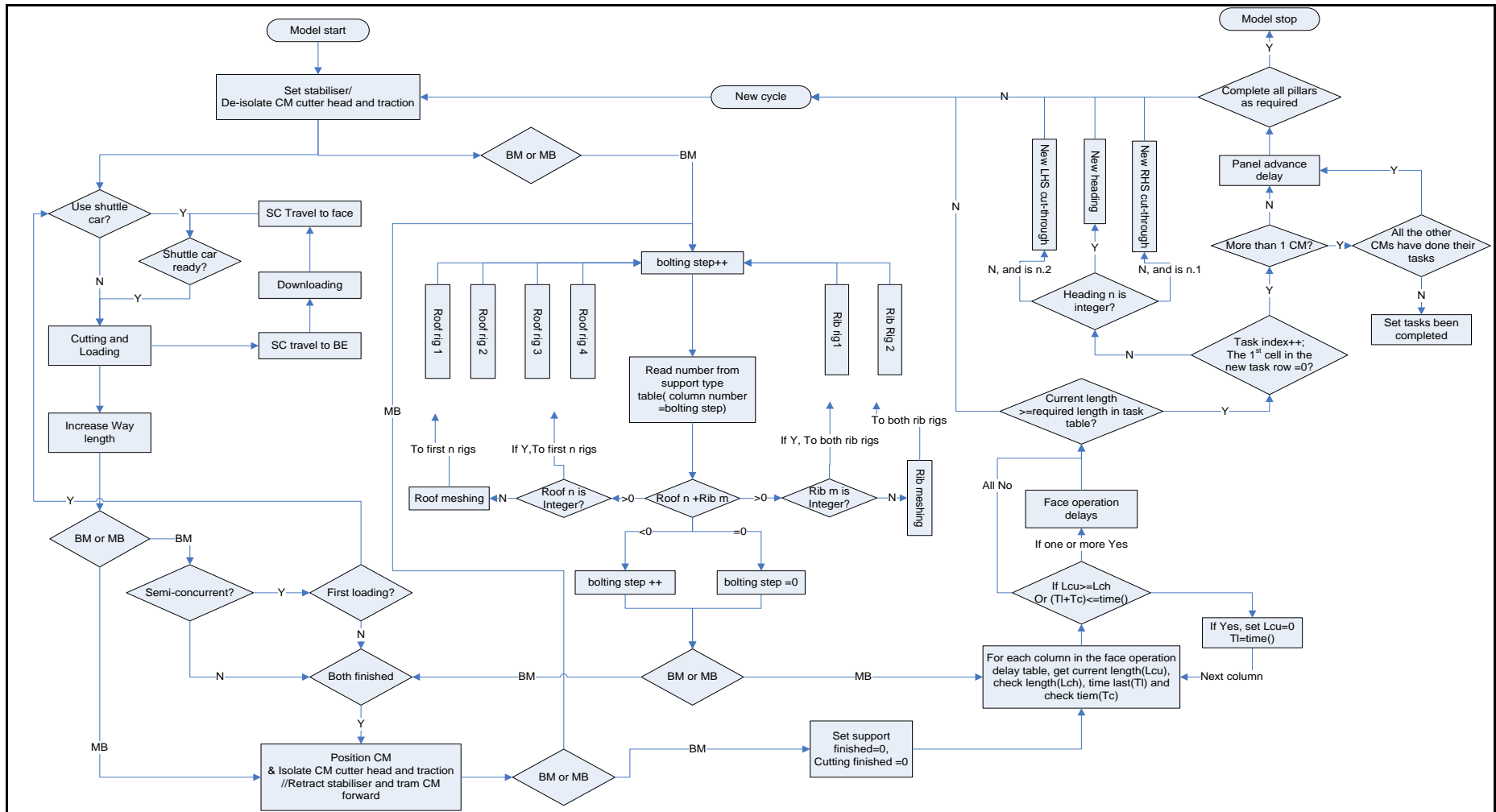


Figure 6-38 Integrated flow chart of the Flexsim roadway development model

6.3 The GUI design

To ensure the model is easy to use, a customised setup GUI was designed for each object, the CM, the SC, the BE and the roadway. The GUIs were designed using the same method discussed in Chapter 6 - the longwall model and were then stored in the variable nodes of each object. They can be called up by a double click on the 3D object. Figure 6-39 shows an example of the CM setup GUI tree node. Only some of the key points have been discussed here.

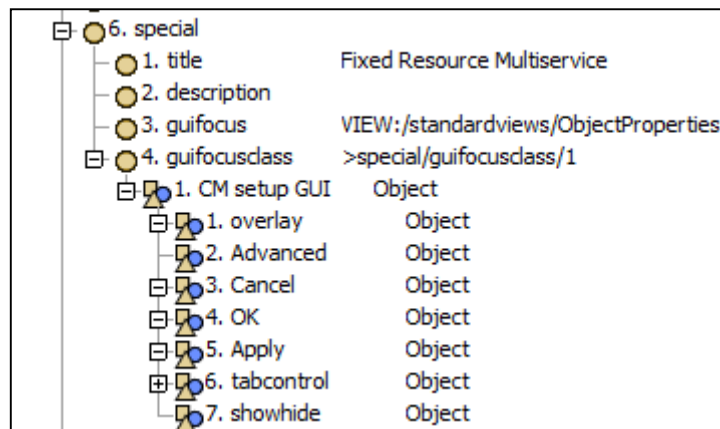


Figure 6-39 The tree structure of the CM setup GUI

There are seven objects in the GUI:

- **overlap**: the default base window of the GUI;
- the **Advanced** button: to open the default properties window of a *processor*;
- the **Cancel**, **OK** and **Apply** buttons;
- **Tab control**: several tab pages are arranged in the **tab control**;
- a **show/hide** button: to show or hide the sub-parts of the CM.

Apart from the **show/hide** button, all the items were shared by all the objects' GUIs, the only differences being the title, the links and the pages (Figure 6-40) of the **tab control**.

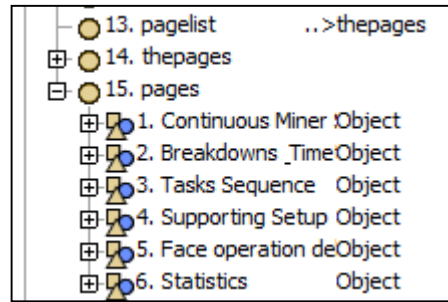


Figure 6-40 The variable treenode of the CM setup GUI tab control and the tab pages

There were six tab pages in the CM setup GUI (Figure 6-40) with each having unique *Flexscript* nodes in each page variable treenode which are triggered when the GUI is open or when the **Apply** button is clicked:

- *PageOnOpen*: to initiate the values on the page view when first opened;
- *PageOnApply*: to apply the setups when the Apply button is clicked on;
- *Refresh*: to update the view whenever the page is refreshed or opened.

Under the variable tree of the GUI root node, there are two *Flexscript* nodes, *OnOpen* and *OnApply*. They are the same as in the default windows of the Flexsim object and were used to run the *Flexscript* nodes in each page when the GUI is open or applied.

6.3.1 The shared tab page

The Breakdowns Timetable page (Figure 6-41) is shared by the CM, the SC and the BE object and is the same as the Breakdown page of normal Flexsim objects.

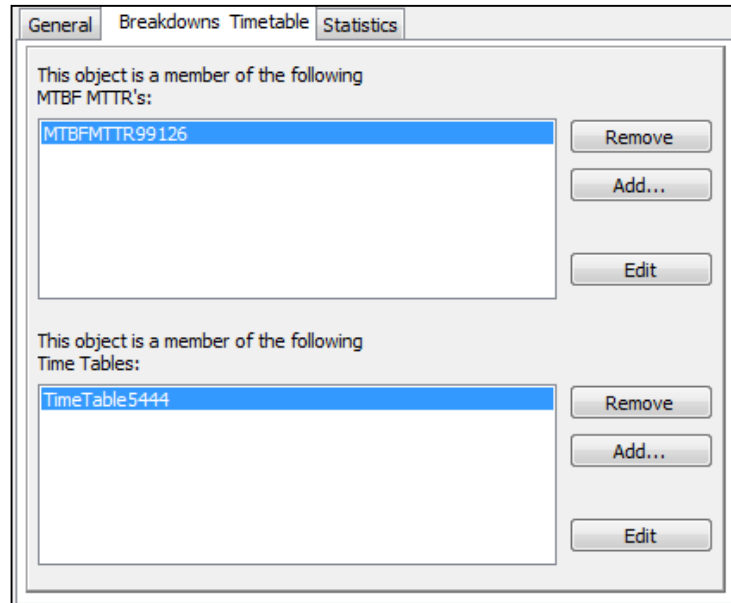


Figure 6-41 The Breakdowns Timetable page

6.3.2 The setup GUI of the Continuous Miner (CM) object

The CM was the main object for a roadway development model. The CM setup GUI contains five pages besides the *Breakdowns Timetable* page, namely *general setup* page (Figure 6-42), *tasks sequence page*, *support setup* page, *face operation delays setup* page and a *statistics* page.

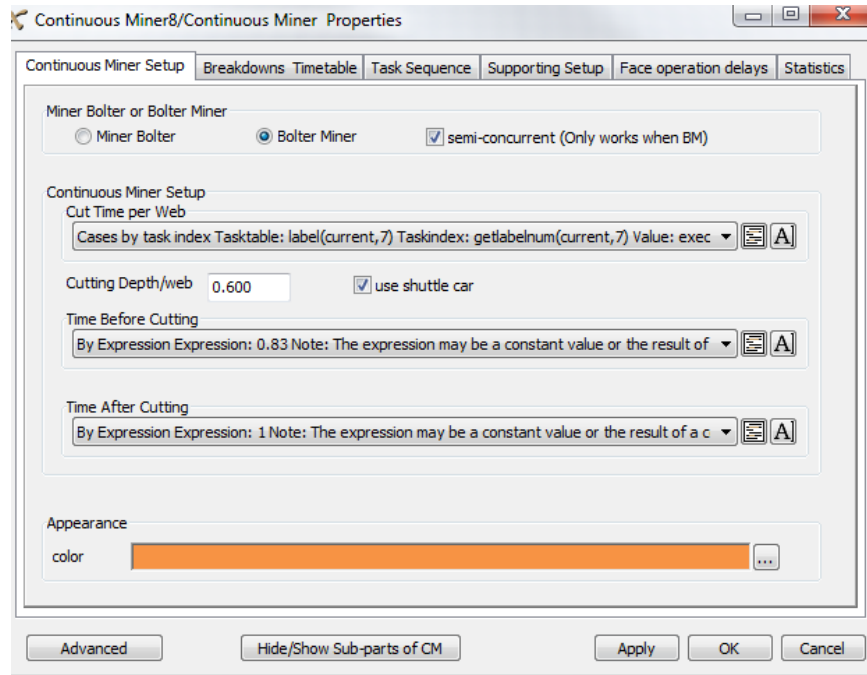


Figure 6-42 The general setup page of the CM

6.3.2.1 The general setup page

In the *general setup* page (Figure 6-42), the user can set up the miner type - either a miner bolter or bolter miner. If it is a bolter miner, the cutting sequence is concurrent or semi-concurrent, while other parameters such as the cut time per web, the cutting web depth, the delay time before and after cutting, the appearance colour of the CM and whether or not to use a shuttle car, can be set up. The miner type setup options are linked to the corresponding label values and are changed by the codes written in the *OnPress* attribute node. The semi-concurrent option has a *coldlink* attribute which is linked to a label.

The three drop down options of the CM setup were modified from the default process time list and linked to the process time variable of each object by setting the *pickprimary* attribute. For example: the *pickprimary* attribute of the process time per web (Figure 6-43) is set to be “@>objectfocus+./../before_cut>variables/cycletime”. By default the cut time per web was set to the value defined in the task sequence table

corresponding to the task index. If the time is the same for all the tasks, the option can be changed to *By Expression* and have a fixed value or a statistical distribution.

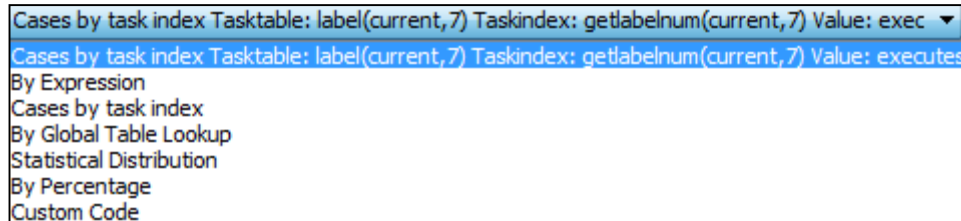


Figure 6-43 The dropdown list of the cut time per web options



6.3.2.2 The tasks sequence setup page

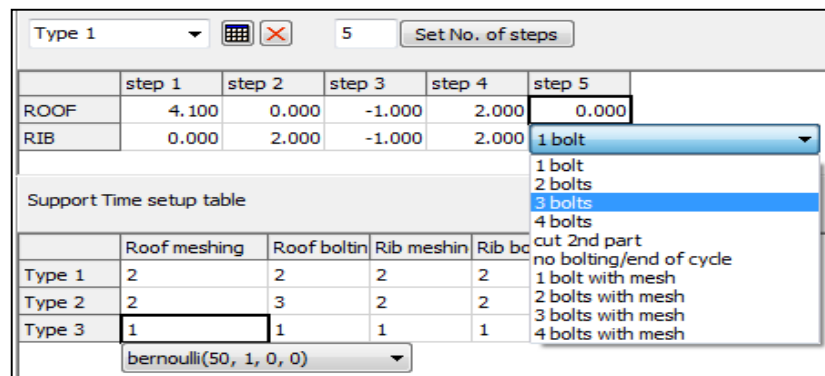
As discussed in the CM tasks sequence section, the tasks table (Figure 6-44) was used to define the development tasks of a CM. A pillar development can be divided into several tasks, with each task having its own parameters. The number of tasks can be set by setting the rows of the table, with the last row ending the task. The table can be used to configure a development sequence of a pillar by simply inputting numbers and selecting from the options. When the first and last column is clicked, a statistics distribution drop down list appears with the same function that was discussed in the longwall model chapter.

Figure 6-44 The tasks sequence setup page

6.3.2.3 The support type and time setup page

The support type and time setup page has three parts (Figure 6-45):

- pick list of support types and steps edit: a *picklist* control to select the types.
Use the button  to add one new type or the button  to delete the selected type. The setup of support type steps is actually to set the size of the table by changing the number of columns;
- support type table: the table is the same as the label in the support cycle except that this table has two drop down lists when any cell of the table is clicked. The maximum number for the bolter to operate at one step (operate in parallel) is four bolts for the roof row and two bolts for the rib row;
- operating time configuration table: this table contains the time for each operation of all the support types. The time can be a constant or a statistical distribution.



	step 1	step 2	step 3	step 4	step 5
ROOF	4.100	0.000	-1.000	2.000	0.000
RIB	0.000	2.000	-1.000	2.000	

	Roof meshing	Roof boltin	Rib meshin	Rib bc
Type 1	2	2	2	2
Type 2	2	3	2	2
Type 3	1	1	1	1

Figure 6-45 The support type and time setup page

When a support type is selected from the drop down list on the top left corner, the link of the support type table is redirected to the selected type and the view is refreshed. This was achieved by codes written in the *OnSelect* attribute node and the “*refreshlist*” Flexscript node.

6.3.2.4 The face operation delays editor

The face operation delays editor (Figure 6-46) is the setup view in the **face operation delays** page. Each delay corresponds to a column in the face operation table. The **Add** button is used to add a copy of the highlighted delay to the list on the left and the parameters then can be edited on the right of the panel.

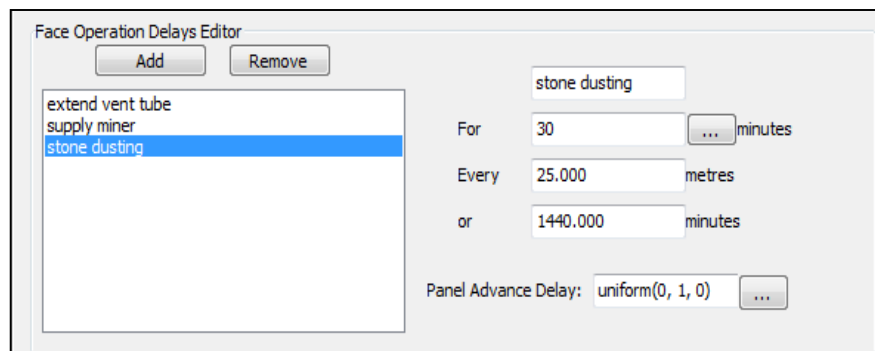


Figure 6-46 The face operation delays editor

In the page root variable tree, two *Flexscript* nodes were created: “*refreshlist*” node to refresh the list and “*refreshdata*” node to refresh the data in the edit boxes when any change is made. The two nodes are executed when a new delay is added, when a delay is selected and when any information about a delay is changed.

6.3.2.5 The statistics page

The statistics page (Figure 6-47) displays the throughput of completed roadway, the support material used and the current state of the CM object. A pie chart (Figure 6-48) of state profile is shown through the **Chart** button.

Throughput

Total Completed Roadway: 0.000

Supporting Material Used

Rib Mesh:	0.000	Roof Mesh:	0.000
Rib Bolt:	0.000	Roof Bolt:	0.000

State

Current: idle

Chart...

Figure 6-47 The statistics page

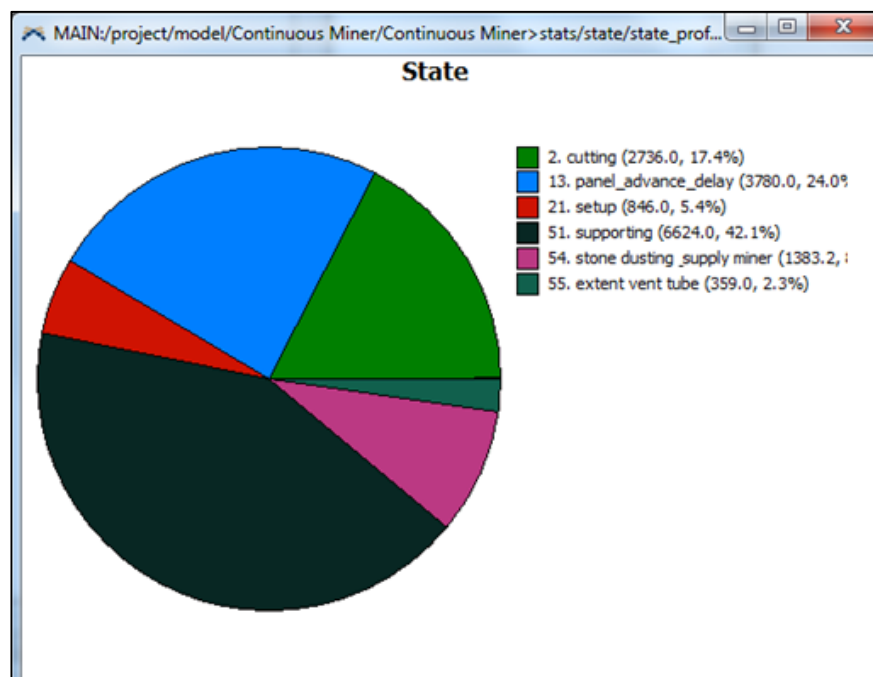


Figure 6-48 The pie chart of the continuous miner state profile

6.3.3 The setup GUI of the roadway object

The setup GUI of the roadway object (Figure 6-49) was used to set up the geometric parameters and the appearance configuration of the roadway, which are linked to the labels of the roadway object.

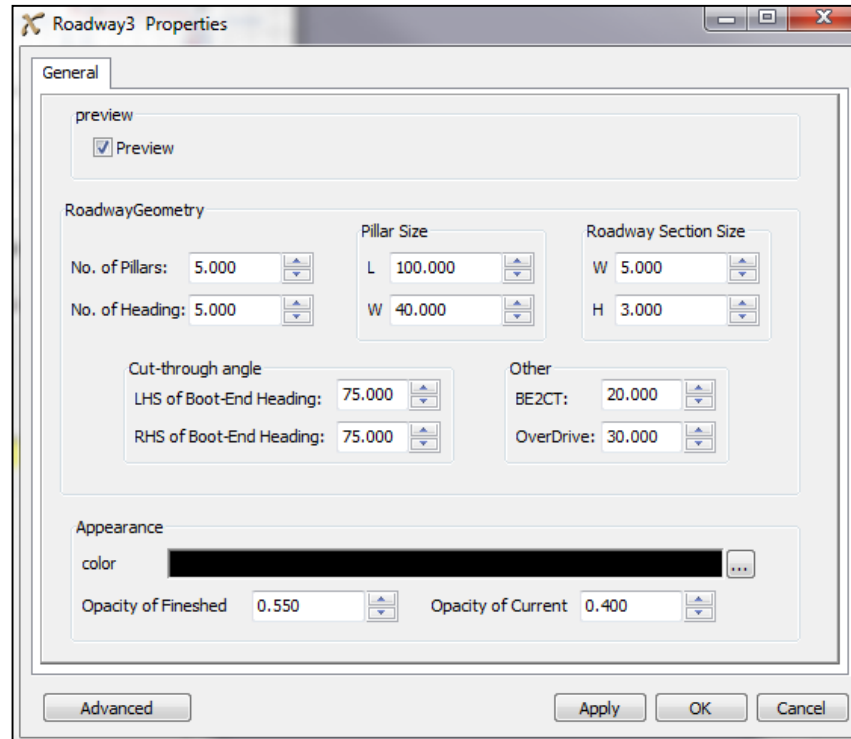


Figure 6-49 The setup GUI of the roadway object

The edit boxes were linked with the labels by adding a *hotlink* attribute to the variable tree of each edit and setting the path referring to the corresponding label. A *hotlink* attribute was used to link a control with a value in the model. The preview checkbox was only used when the model user wanted to preview the pillars when configuring the model. The checkbox was linked with a flag value stored in a label of the roadway object by the *OnPress* attribute and whenever the checkbox was clicked, the label value was changed to a value that was opposite the current value. For example, it changed to 0 when the current value was 1 on one click and then changed back to 1 after another click. The codes are as follows:

```
treenode preview = node("@>objectfocus+>labels/preview",c);
setnodenum(preview, !getnodenum(preview));
```

When the page is open, the following codes in the *PageOnOpen* attribute node of the general page is executed to initiate the checkbox:

```
treenode preview = node("../3/1",c);
setchecked(preview, getlabelnum(focus,2));
```

6.3.4 The setup GUI of the boot end (BE) object

The BE object has only one parameter to configure, apart from the size and colour, which is the heading location of the BE (Figure 6-50). The BE is then automatically placed at the correct location by codes when the model is reset. The size and colour can also be changed but this does not affect any aspect of the simulation except the appearance of the BE object.

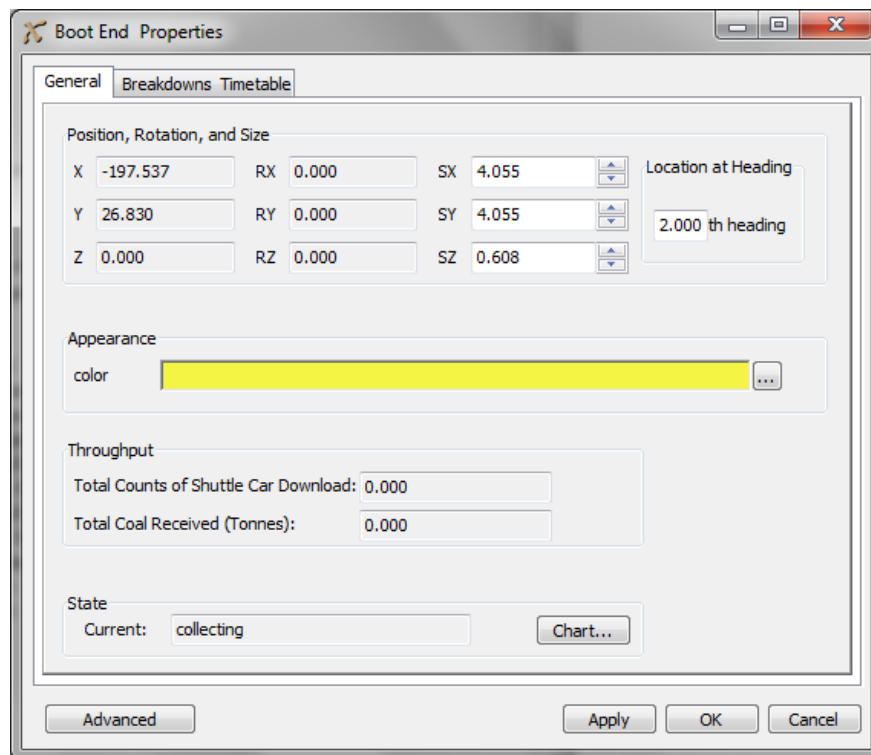


Figure 6-50 The setup GUI of the BE object

At the bottom part of the *General* page, some statistics are shown, as are the total counts of SC downloads, total coal received and the state of the BE.

6.3.5 The setup GUI of the Shuttle Car (SC) object

The setup GUI of the SC object (Figure 6-51) has three tab pages: *General*, *Breakdowns Timetable* and *Statistics*. The *Breakdowns Timetable* is a shared page, as discussed previously, while the *Statistics* page has items that are similar to the bottom part of the BE *General* setup page.

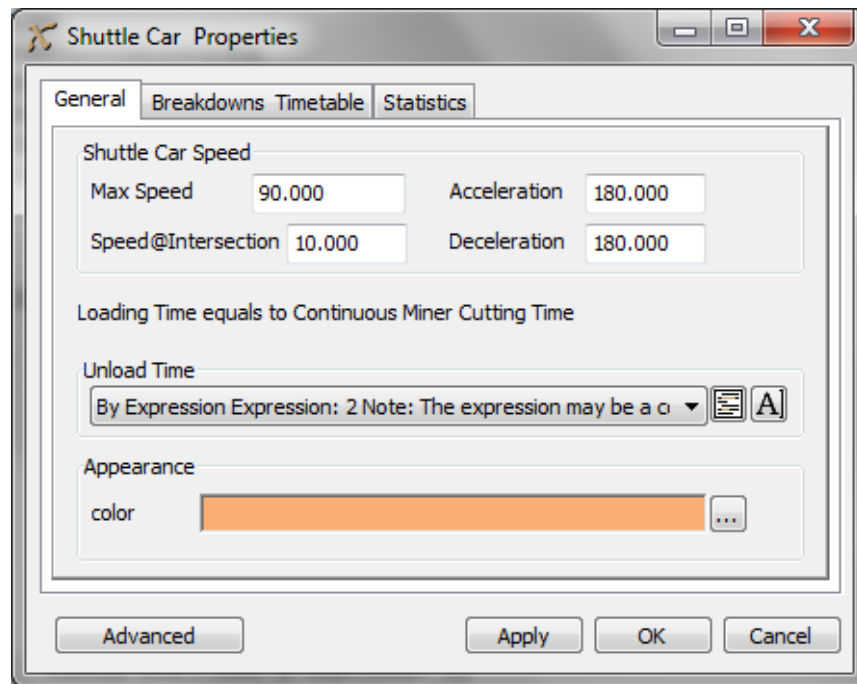


Figure 6-51 The setup GUI of the SC object

The *General* page has two main parts, the speed setup and the unload time setup. The speed setup includes the SC max speed, speed at roadway intersections and acceleration and deceleration. The speed at the roadway intersections is the speed when the SC travels through the intersection of the heading and the cut-through and this value is stored in a label of the SC object. The other three speed parameters are default variables of the *Task Executer* object (the SC). The unload time setup is the default *picklist* control of a *Task Executer*. Actually, the *Task Executer* object also has a *picklist* option to set the time to load but this value has been set at 0 and the model user cannot access it

through the customised GUI because the load time of a SC is assumed to be equal to the CM cutting time.

6.4 User object library

Unlike the longwall model where all the essential items of equipment were grouped together and made into one unit, the four fundamental equipment items for roadway development were designed separately. New objects can be added to the user library by right clicking on the 3D objects and selecting **Edit | Add to User Library**. The final roadway development objects user library is as shown in Figure 6-52, including CM, SC, BE and roadway layout with prewritten scripts.

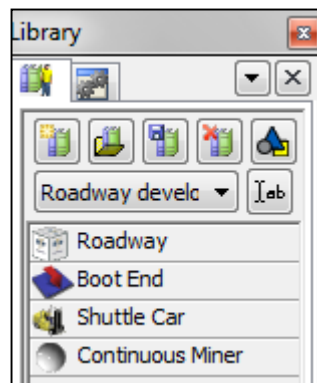


Figure 6-52 User library of roadway development objects

6.4.1 Building models with the user library

The objects in the user library act as normal Flexsim objects and can be added into a model using the drag and drop method. To build a model, drag the required objects and drop them into the 3D working environment and then connect the objects by following the rules below:

- press the A key to connect the CM to the BE. This is a normal connection that indicates coal flows from the CM to the BE;

- press the S key (centre) to connect the CM with the SC. This SC now serves the CM and transports coal from the CM to the BE if the CM is configured to use the SC;
- press the S key (centre) to connect the BE with the roadway. This is for referring objects by prewritten scripts.

Only two SCs can be added for each CM, but each BE can have more than one CM. After completing all the drag and drop procedures and all the required connections, a model with a default configuration can then be built. To change the configuration, double click on the 3D object to show the customised GUIs or use the **Advanced** button to access the default properties window. An example of a model view is shown in Figure 6-53.

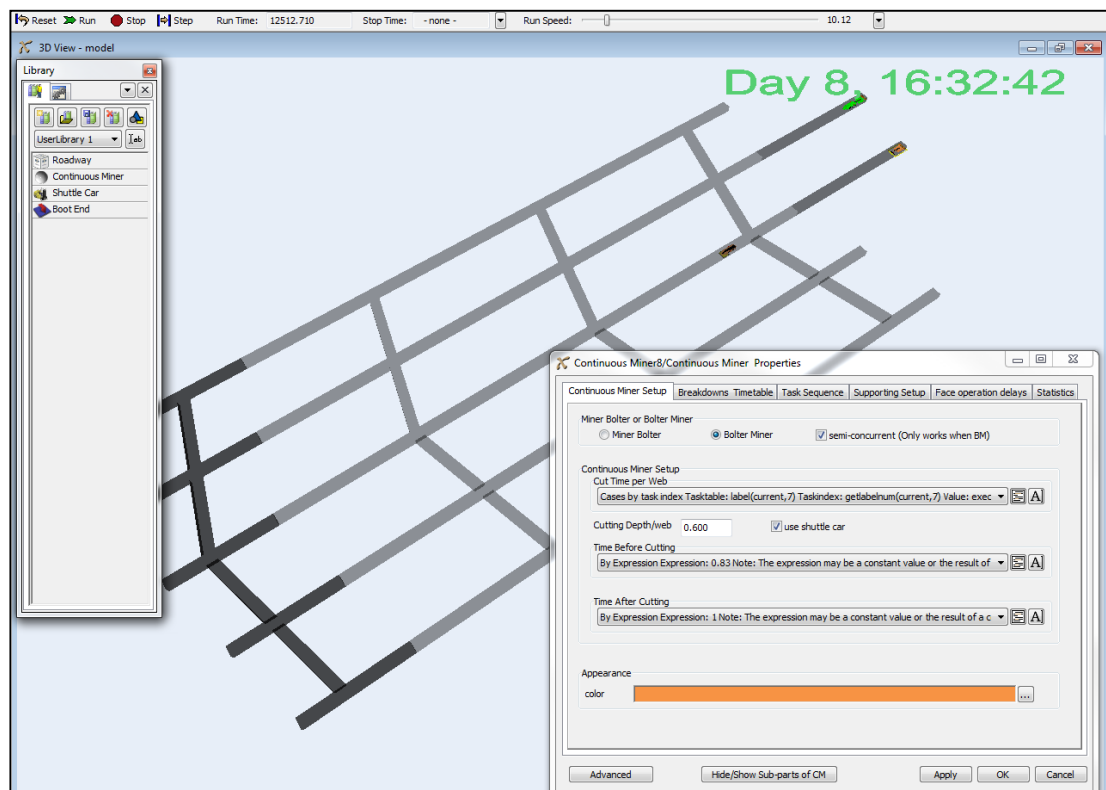


Figure 6-53 The final view of a model

Figure 6-54 shows four scenarios using different haulage strategies using the library.



Figure 6-54 The scenarios of using different haulage strategies

6.5 The features of the roadway development simulation model

The simulation model has the following features:

- the model is object oriented with 3D objects. It performs in a virtual reality environment and what can be seen from the model is what really happens. Each object can work as a real piece of roadway development equipment or together as a roadway development system;
- the configurable and structured model design with optional strategies and user friendly GUIs makes the model easy to use and advanced knowledge of Flexsim® is not required;

- the model can simulate most standard roadway layout by changing the values of the width and length and it is flexible enough to simulate multi-heading panels by using the task sequence table;
- the model can simulate different miner types and complicated development sequences by configuring a numerical table;
- the model is flexible enough to configure any face operation delays;
- the design of the support cycle makes it possible to configure any standard support type and multiple support types can be applied in different sections of one pillar;
- multiple haulage strategies such as one SC, two SCs or CHS can be applied and real time SC interactions can be simulated;
- multiple scenario experiments can be performed using the integrated analysis tool of Flexsim®.

The simulation model has the following limitations in its current form:

- the shape of the roadways must be in repetitive blocks. The model only repeats one pillar development and all the configurations such as pillar size and development task sequence, are based on one pillar, except the downtime which may be collected over a long period of time;
- the model cannot simulate the place change method of roadway development. At present it can only simulate the in-place method using CMs with mounted rigs.

6.6 Model validation

As with the longwall model, the roadway development model can perform a single run to examine the performance of a given configuration or use the Experimenter to evaluate and compare multiple scenarios with several changing variables.

Two separate companies' data were used to validate the roadway development model and the base configuration of the sensitivity analysis. Their historical data comprised:

- roadway geometry data;
- process parameters;
- cycle times;
- historical delays information; and
- development rates achieved.

6.6.1 Mine 2 scenario

As mentioned in Chapter 5, the development units of Mine 2 were based on a two heading configuration using one bolter-miner supported by one shuttle car. The pillars were 102 m long and 40 m wide. The roof and rib conditions required a bolting density of six roof bolts and three rib bolts on each rib every 0.92 metres, with an average time to complete the support operations of 12 minutes.

6.6.1.1 Basic data input

Table 6-4 gives the basic configuration data of Mine 2, including the roadway geometry, machine capacities, miner type and face operations. However, the data was modified to fit the model input and some of the data was estimated within a relatively short period of onsite observation, for example the SC speed and the time per web.

Table 6-4 Basic configuration data of Mine 2

	Units	Value
Pillar Layout		
Pillar length	m	102
Cut-through length	m	40
Overdrive	m	20
Cut-through to Boot End	m	20
Heading width	m	5.2
Mining height	m	3.4
Continuous Miner		
Type	Bolter-Miner	
Web depth (One load of SC)	m	0.46
Time per web (cut and load one SC)	min	1.74
Shuttle Car		
Car capacity	t	13
Discharge time per one load	min	1.74
Max wheeling speed	m/min	95
Number of loads in each support cycle	#	2
Support Operations		
Type	Concurrent	
Total bolting time	min	12
Boot End		
Out-bye conveyor rate	Unlimited	
Face Operations		
Install vent tubes every	m	4
Install vent tubes duration	min	4
Stone dusting every (distance)	m	25
Stone dusting every (time)	h	24
Stone dusting duration	min	30
Supply CM every	m	15
Supply CM duration	min	15
Other		
Panel advance	h	24
Coal density	t/m ³	1.5

In the Flexsim model the cutting web depth was set at 0.46 m which is half the advance in one cycle (two SC loads). The total amount of coal per one web was 12.12 tonnes resulting from the web depth, size of the roadway section and density of the coal. The support cycle was divided into two steps, with each step having six minutes of roof bolting time. The support type was set to be 4, -1, 4 and 0 in the roof row of the support

type table and set to be 0, -1, 0 and 0 in the rib row. This set up was only to simulate the 12 minutes total support time in one cycle, regardless of the number of bolts. The miner type was set to be BM with a concurrent cutting sequence which means the cutting and loading process only started when both SCs were ready and half the support cycle was completed. The cutting processing time depended on the amount of coal and the SC loading rate, which was about 1.74 minutes on average. The SC unloading time was also 1.74 minutes. All the other values were set as shown in the table, so in each cycle the sequence was:

- i. cut and load one SC, which lasted for about 1.74 minutes and when it finishes loading, the SC left to go to the BE;
- ii. at the same time the cutting and loading started, the support cycle began and lasted for 6 minutes;
- iii. when the SC returned and the support cycle was completed, the other half of the cycle began.

6.6.1.2 Development sequence setup

The development sequence of Mine 2 is shown in Figure 6-55 and is as follows:

- i. develop heading A from the end of the last overdrive for 102 m;
- ii. the miner flits back for 20 m and starts to cut the cut-through for 40 m (actually 35 m as the distance is from centre to centre and the roadway width is 5 m);
- iii. the miner flits back to the end of last overdrive of Heading B and cuts for 102 m.

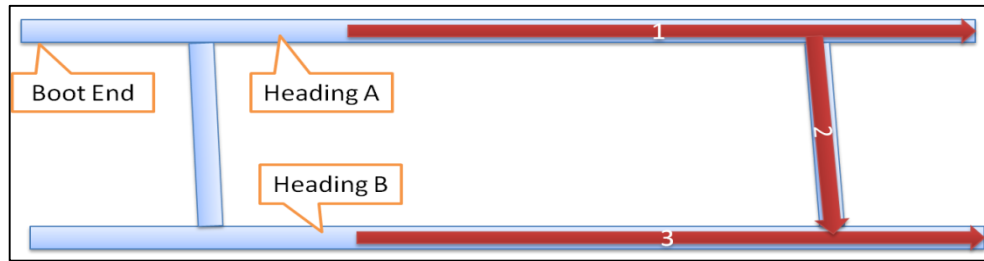


Figure 6-55 Development sequence of Mine 2

The setup of the development sequence in the Flexsim model is shown in Figure 6-56. From task 1 to task 2 and task 2 to task 3, there should be some delays moving the working unit to the new location, extra time to cut the break away and extra time of holing through. However, these delay times were not separated in the historical data and was probably included in the down time, so in the task sequence table, they were all set to be 0.

	time/web	frompoint	topoint	heading	supporttype	delay2nesktask
Task 1	1.74	20.00	122.00	1.00	1.00	0
Task 2	1.74	0.00	35.00	1.10	1.00	0
Task 3	1.74	20.00	122.00	2.00	1.00	0
End of task	0	0.00	0.00	0.00	0.00	0

Figure 6-56 The development (task) sequence setup of Mine 2

6.6.1.3 The MTBF/MTTR and shift schedule time table

The delay data of Mine 2 is as shown in Table 6-5. All the categories of the delays fully stop the model from producing.

Table 6-5 The delay data of Mine 2

Delay Category	Total Duration (% of Operating Time)	Delay Duration (min)			Mean Time Between Events (hours)
		Min	Avg	Max	
Mine Process Unplanned	0.2%	49	50	51	480
Outbye Services Unplanned (short delay)	9.0%	10	36	70	6
Outbye Services Unplanned (long delay)	18.7%	150	530	900	44
Panel Engineering Unplanned	6.3%	10	46	150	11.4
Panel Process Unplanned	5.0%	10	51	100	16

Mine 2 operated on a shift schedule consisting of

- three 10 hour shifts per day Monday to Thursday;
- two 12 hour shifts were scheduled for Fridays; while a single
- 12 hour shift was scheduled for Saturdays and Sundays;
- a total of 15 hours of planned maintenance was scheduled per week.

And

- 10 hour shifts have a one hour break at the 5th hour;
- 12 hour shift have two 45 minute breaks every 4th hour; except the last
- one hour delay before starting work and one hour delay after finishing work.

So the 10 hour shift had only 7 hours working time and the 12 hour shift had only 8.5 hours working time to operate the CM and SC. A table consisting of a one week plan was designed and configured as shown in Figure 6-57. The table has 49 rows and repeats every 10080 minutes (one week). As can be seen, the table has many rows with a duration of 0.01 minutes. This is the time when a new shift starts to work and the duration was used to record this event. The codes were written in the Down Function and Resume Function.

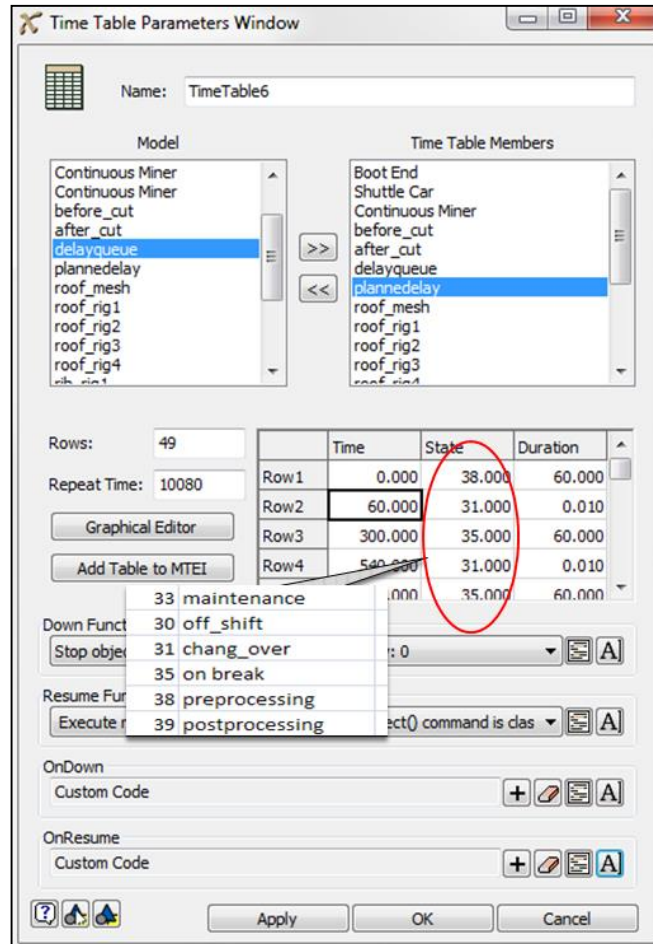


Figure 6-57 The time table configuration of Mine 2 shift schedule

6.6.1.4 Model reports

The model was then run for 10 replications. Each replication simulated 20 pillars but with a different seed in generating the random occurrences of unplanned delays. A comparison of average key performance indicator (KPI) results of the Flexsim model and the historical data provided by the coal mine were summarised in Table 6-6.

Table 6-6 KPI comparison between Flexsim output and historical data of Mine 2

	From Mine	Flexsim Output
MODEL OUTPUT		
Development Rate (days/Km)	108	105.9
First Coal - First Coal (days)	11	10.8
Confidence interval (days +/-)	--	0.3
Pillar Advance (days)	1	1
Advance rate		
m/planned hour	1.8	1.8
m/operating hour	4	3.6
Miner Utilisation (%)		
Scheduled Production Time	61	61.5
Planned Process Time	52	52.2
Operating Time	--	25.9
Net Cutting Time	--	7.9

The Flexsim model predicted an average pillar cycle time of 10.8 ± 0.3 days compared to an historical performance of 11 days at Mine 2. The results of the advance rates were the same in m/planned hour which was 1.8 and similar to m/operating hour which was 3.6 compared to 4.0.

When considering utilising the miner, the scheduled production time is the total calendar time minus the time when no labour was scheduled and the planned process time is the scheduled production time less the planned delay, which was the panel advance time in this model and then the operating time was the planned process time less breakdown time which was about 36.3% of total calendar time. The model predicted almost the same scheduled production time and planned process time and also predicted the operating time and net cutting time which had not been included in the historical data. As can be seen, the operating time was only slightly more than a quarter of the total calendar time while the net cutting time was only 7.9%. It follows that every

percentage of calendar time that can be changed from other operations or delays to the cutting process results in a major increase to the advance rate.

Before the previous experiment, the initial validation involved simulating the development of between one and thirty pillars to determine the minimum number of contiguous pillars required to produce a stable simulation. This initial validation was required as the 'Shift Scheduler' and MTBF/MTTR within Flexsim[®] resets at the end of each replication. If less than a minimum number of pillars were modelled then the effect of long weekend shifts and breakdowns could not be taken into account accurately or add much variation to the results. It was found that a minimum of 20 pillars were required to produce a stable model (Figure 6-58).

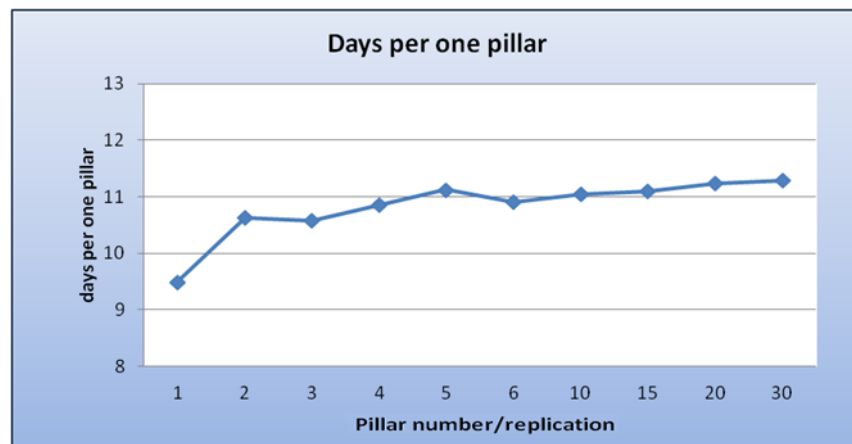


Figure 6-58 Days per pillar versus a range of pillar numbers per replication for Mine 2

Comparative histograms of the actual and simulated metres cut per shift at Mine 2 are provided in Figure 6-59. The model predicted the same range of metres cut per shift and the frequencies of metres cut per shift were similar to the most frequent shift advance rates of around 10 to 20 metres.

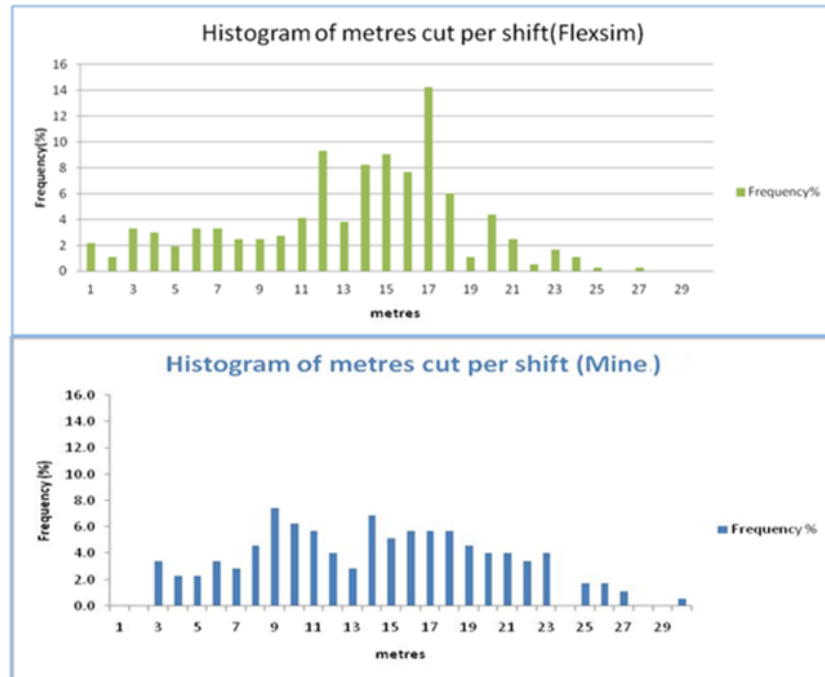


Figure 6-59 Histograms of Mine 2 simulated and actual metres cut per shift

6.6.2 Mine 3 scenario

As mentioned in Chapter 5, roadway development operations at Mine 3 were based on a two heading configuration using two miner-bolters each supported by one shuttle car. The pillars were 125 metres long and 45 metres wide. The roof and rib conditions required a bolting density of six roof bolts and three rib bolts on each rib every 1.2 metres, with an average time to complete the support operations of 22 minutes.

6.6.2.1 Basic data input

Table 6-7 gives the basic configuration data of Mine 3, including roadway geometry, machine capacities, miner type and face operations.

Table 6-7 Basic configuration data of Mine 3

	Units	Value
Pillar Layout		
Pillar length	m	125
Cut-through length	m	45
Overdrive	m	20
Cut-through to Boot End	m	30
Heading width	m	5
Mining height	m	3
Continuous Miner		
Type	Miner-Bolter	
Web depth (One load of SC)	m	0.6
Time per web (cut and load one SC)	min	2.52
Shuttle Car		
Car capacity	t	13
Discharge time per one load	min	1.01
Max wheeling speed	m/min	80
Number of loads in each support cycle	#	2
Support Operations		
Type	Noncurrent	
Total bolting time	min	22
Boot End		
Out-bye conveyor rate	Unlimited	
Face Operations		
Install vent tubes every	m	2.4
Install vent tubes duration	min	2
Stone dusting every (distance)	m	30
Stone dusting every (time)	h	24
Stone dusting duration	min	30
Supply CM every	m	30
Supply CM duration	min	30
Other		
Panel advance	h	22
Coal density	t/m ³	1.4

The major difference between Mine 2 and Mine 3 was that Mine 3 used two MBs while Mine 2 used one BM. The support cycle has the same setup as Mine 2, but the roof bolting time was set at eleven minutes instead of six minutes.

6.6.2.2 Development sequence setup

The development sequence of Mine 3 is shown in Figure 6-60. The CM2 at Heading B starts to cut Heading B from the end of the last overdrive while CM1 follows the sequences as follows:

- i. develop heading A from the end of the last overdrive for 105 m;
- ii. cut the cut-through for 45 m (actually 40 m as the distance is from centre to centre and the roadway width is 5 m);
- iii. miner flits back into Heading A and continues to cut the overdrive for 20 m.

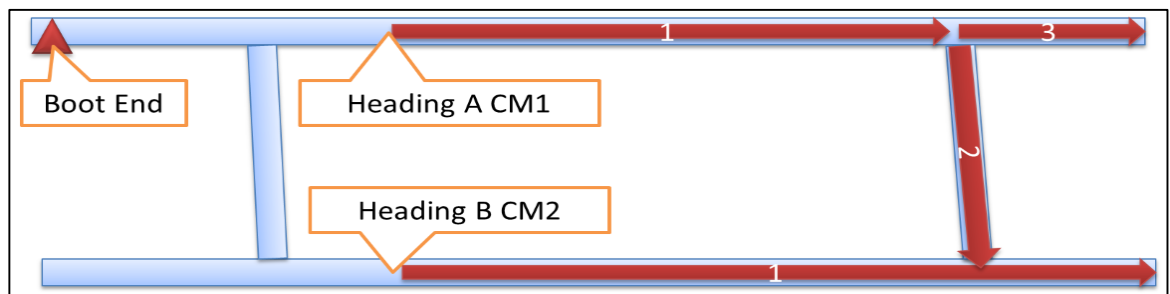


Figure 6-60 Development sequence of Mine 3

6.6.2.3 The MTBF/MTTR and shift schedule time table

The delay data of Mine 3 is as shown in Table 6-8. All the delay categories fully stop two CMs from producing coal and supporting the roadway.

Table 6-8 The delay data of Mine 3

Delay Category	Total Duration (% of Operating Time)	Delay Duration (min)			Mean Time Between Events (hours)
		Min	Avg	Max	
Mine Process Unplanned	2.3%	180	200	240	144
Outbye Services Unplanned	6.7%	15	45	180	10.4
Panel Engineering Unplanned	8.5%	10	52	120	9.3
Panel Process Unplanned	7.1%	10	48	92	10.5

Mine 3 operated on a shift schedule consisting of:

- three shifts per day Monday to Friday. The day and night shifts lasted for 10 hours, whilst the afternoon shift lasted 8.5 hours;
- two 12 hour shifts were scheduled for Saturdays and Sundays;
- A total of 17 hours of planned maintenance was scheduled per week (one 12 hour shift on Saturday and one 12 hour shift on Sunday) and
- 10 hour shifts have a one hour break at the 5th hour;
- 12 hour shifts have two 45 minute breaks every 4th hour except the last;
- One hour delay before starting work and one hour delay after finishing work.

The same method as Mine 2 was used to configure the timetable as a one week shift plan for the Mine 3 scenario.

6.6.2.4 Model reports

An initial validation involved simulating the development of between 1 and 32 pillars to determine the minimum number of contiguous pillars required to produce a stable simulation. As Figure 6-61 shows, when the number of pillars in each replication is more than 4, the model can produce a stable result of days per one pillar. When less than four pillars were modelled the apparent rate of development dropped dramatically because the shift pattern did not have enough weight in the simulation and some long term breakdowns may not be taken into account.

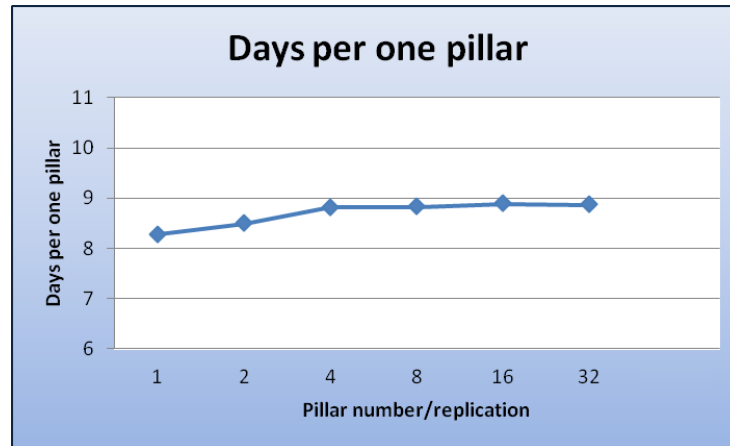


Figure 6-61 Days per pillar versus a range of pillar numbers per replication of Mine 3

The model was run for ten replications and each replication of the model simulated ten pillars. A comparison of the average KPI results of the Flexsim model and the historical data provided by Mine 3 are summarised in Table 6-9. The Flexsim model predicted an average pillar cycle time of 8.9 ± 0.4 days compared to an historical average of nine days at Mine 3. The advance rate predicted by the model was slightly higher at 1.2 m/planned hour than the historical data of 1 while 1.9 compared to 1.8 using the measure of m/operating hour.

Table 6-9 KPI comparison between Flexsim output and historical data of Mine 3

	From Mine	Flexsim Output
MODEL OUTPUT		
Development Rate (days/Km)	72	71.2
First Coal - First Coal (days)	9	8.9
Confidence interval (days +/-)	--	0.4
Pillar Advance (days)	0.9	0.9
Advance rate		
m/planned hour	1	1.2
m/operating hour	1.8	1.9
Miner Utilisation (%)		
Scheduled Production Time	70	68.4
Planned Process Time	57	58.1
Operating Time	40	36.3
Net Cutting Time	--	6.3

The miner utilisation in the Flexsim output was an average of the two MBs and the CM1 had a higher net cutting time than the CM2 because it cut fewer roadways than CM1 and had to stop and wait for CM1. The results for the miner utilisation were satisfactory.

The actual and simulated metres cut per shift at Mine 3 were more satisfactory than Mine 2. As Figure 6-62 shows, Flexsim[®] predicted almost the same range and the same frequency of metres cut per shift.

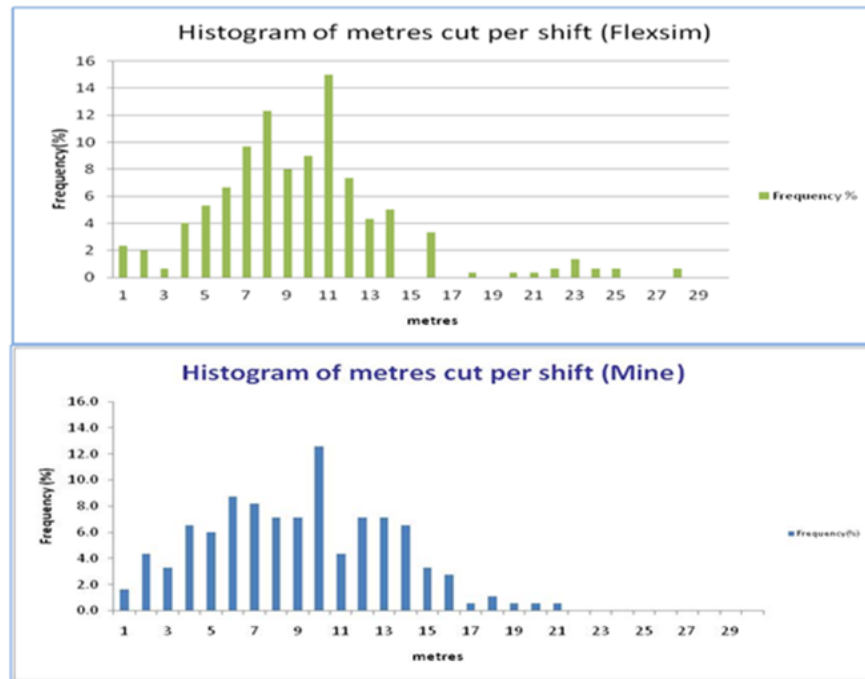


Figure 6-62 Histograms of Mine 3 simulated and actual metres cut per shift

In both case studies the Flexsim[®] simulation results were satisfactory compared to the mines' historical performance, which means the Flexsim[®] model was validated.

6.7 Sensitivity analysis

The validated Flexsim model was then used to assess how changing various aspects of the operations impacted on the development rates when using the same data and mining sequence as Mine 2. A comparison was made when changes were made to the:

- miner type: bolter miner versus miner bolter;
- cycle time for bolting;
- shuttle car tramming speed;
- shuttle car loading time;
- haulage system: shuttle car versus continuous haulage system;
- cutting and loading cycle time of the continuous miner at the development face.

6.7.1 Single impact factor

The following two experiments demonstrate the impact of only one change of the system.

6.7.1.1 Bolter-miner versus miner-bolter

As discussed, this was easy to simulate in the developed Flexsim model by using the options of miner bolter and bolter miner. The results comparing the performance between a bolter miner and a miner bolter using mine 2 data are shown in Figure 6-63.

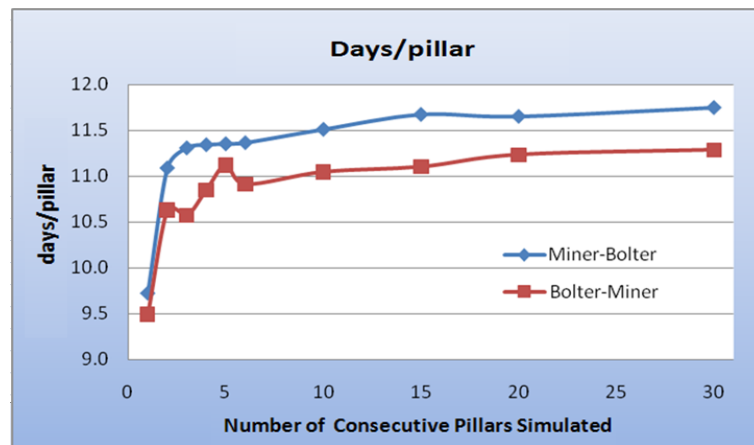


Figure 6-63 Days per pillar of bolter-miner versus miner-bolter

As the Figure 6-63 shows, the BM was about half a day faster per pillar than the MB. This difference was smaller than expected because a BM was expected to give a major increase in the development rate because the cutting and supporting operations are in parallel mode. Numeric calculation shows the average shuttle car away time is about 5 minutes and the corresponding support operation took 6 minutes, the support operation was almost done within the shuttle car away time. A bolter miner only improves the primary cutting, loading and supporting cycle by about one minutes from about 7 minutes, which is 14%. However, the operating time is only about 26% of total calendar

time. The delays and breakdowns cut the expected improvement to about 4%, which is only a small improvement with a BM compared to an MB.

6.7.1.2 Impact of bolting cycle time

Figure 6-64 shows the results of the development rate versus a range of support cycle times from two to eighteen minutes while all other factors remained the same. The support cycle times in excess of 11 minutes had a significant impact on the development rate and implied that if new bolting technology was introduced the cycle time must be less than 11 minutes to ensure that bolting was removed as a bottleneck.

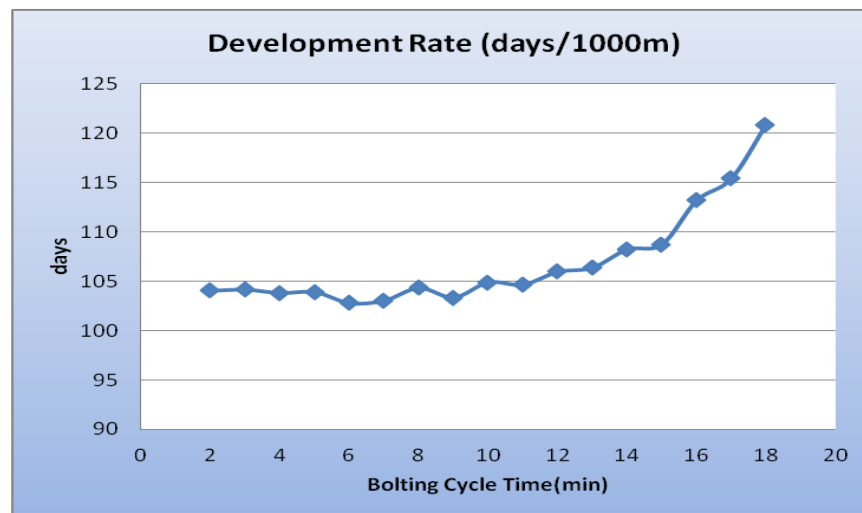


Figure 6-64 Impact of bolting cycle time

As the figure shows, when the bolting cycle time was less than 11 minutes the development rate fluctuated around 104 days/1000 m and when the time was more than 11 minutes, the development rate was slowed down by the support cycle and increased from about 105 days/1000 m to more than 120 days/1000 m. This means that the current support operation was almost optimised by twelve minutes. A faster support operation would not improve the development rate much because the bottleneck was elsewhere,

most probably the SC, because the CM must wait for the shuttle car after the bolting cycle when SC away time is longer than about 4.3 minutes.

6.7.2 Multi impact factors

As shown in the previous experiments, it can be difficult to promote productivity by changing a single aspect of the system. The following experiments respectively show the combined effects on the development rate of:

- SC unloading time and the bolting cycle (support cycle) time;
- SC travelling speed and bolting cycle time;
- CM cutting and loading time and the bolting cycle time.

6.7.2.1 Impact of shuttle car download time and bolting time

Figure 6-65 shows the impact on the development rates in units of days/pillar of a range of shuttle car download times from 0.5 minutes to 4 minutes and a range of bolting cycle times from 10 minutes to 20 minutes. The fastest rate of development was about 9.5 days/pillar at the bottom left corner of the figure with the fastest bolting cycle time and SC download time. When one of the two factors was fast, the other became a determining factor on the development rate, which constrained the development rate if it was very slow because the support operation and the SC away time are mostly in parallel.

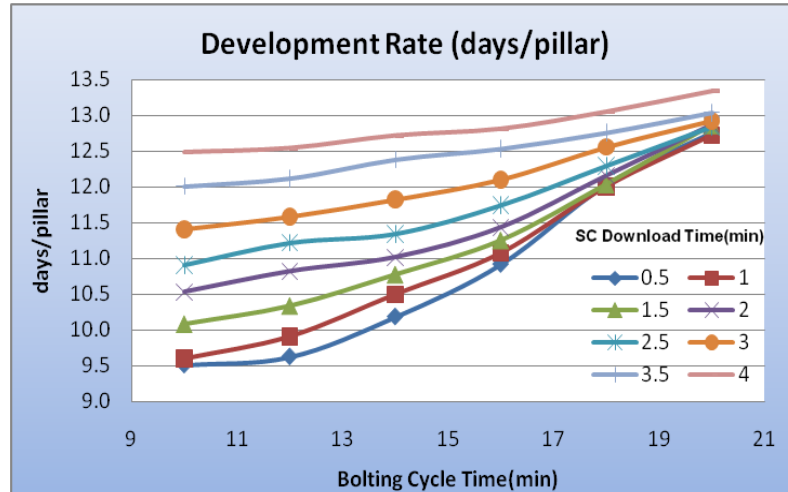


Figure 6-65 Impact of shuttle car download time and bolting time

6.7.2.2 Impact of shuttle car speed and bolting time

Figure 6-66 shows the impact on the development rates in units of days/pillar of a range of shuttle cars maximum travelling speeds from 70 m/minute to 110 m/minute and a range of bolting cycle times from 6 minutes to 22 minutes. This figure shows a similar feature to Figure 6-65, as both the SC download time and the SC travelling time (max speed) are part of the SC away time. The graphs converge when both support cycle and SC travelling speed are slow, but if the bolting cycle is very slow, no matter how fast the SC travels, the development rate does not change.

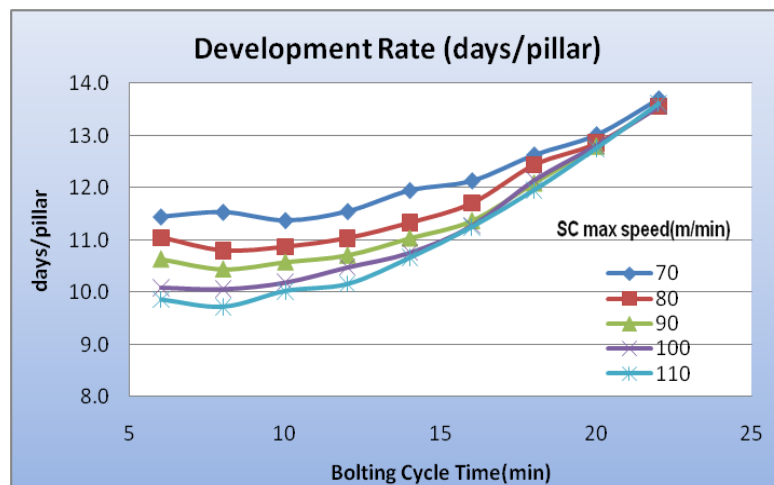


Figure 6-66 Impact of shuttle car speed and bolting time

6.7.2.3 Impact of CM cutting and loading time and bolting time

Figure 6-67 shows the impact on the advance rate in units of m/operating hour of a range of CM cutting and loading times from 0.5 minutes/web to 3.5 minutes/web and a range of bolting cycle times from 10 minutes to 24 minutes. This figure showed a different feature from the previous two scenarios. The graphs had a slow convergence rate and the rate of decrease of the curves stayed almost the same when the bolting cycle slowed. Thus, in Mine 2, the advance rate can be promoted by improving the support operation or the CM cutting and loading process. Since most of the support cycle was done after cutting and loading, the CM acted mostly like an MB which does cutting and supporting in sequence, even though it is a BM.

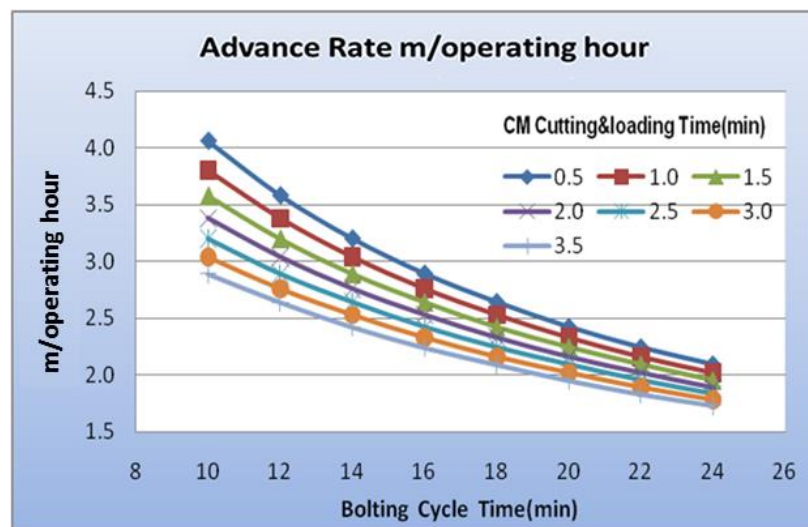


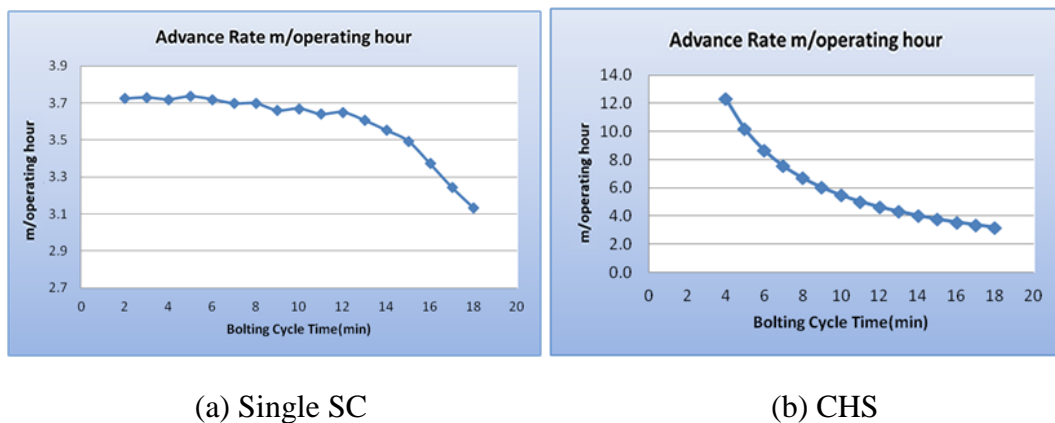
Figure 6-67 Impact of CM cutting and loading time and bolting time

6.7.3 Impact of Continuous Haulage System (CHS)

The previous experiments showed that the fastest advance rate was slightly faster than 4 m/operating hour and the fastest pillar cycle time was always more than 9.5 days. There was no significant improvement to the system by changing the support cycle, cutting and loading cycle and the SC.

Continuous haulage systems, which were expected to make a marked improvement in roadway development performance, have been introduced into underground coal mining for some time but have not been widely used. What would happen if a CHS is introduced into Mine 2?

Figure 6-68 shows that the CHS improved the advance rate considerably. The same CM and working schedule can obtain about 10 m/operating hour or even more when the bolting cycle time is less than 6 minutes which is not difficult to achieve. This is more than twice the rate obtained when using a SC. Across the range of 6 to 15 minutes of bolting cycle time there was a marked but decreasing improvement using a CHS from almost the same advance rate of 3.5 m/operating hour obtained with the SC. However, the CHS made almost no difference when the bolting time was more than 15 minutes compared to using a SC. A CHS was expected to improve roadway development quite considerably when the bolting cycle is fast.

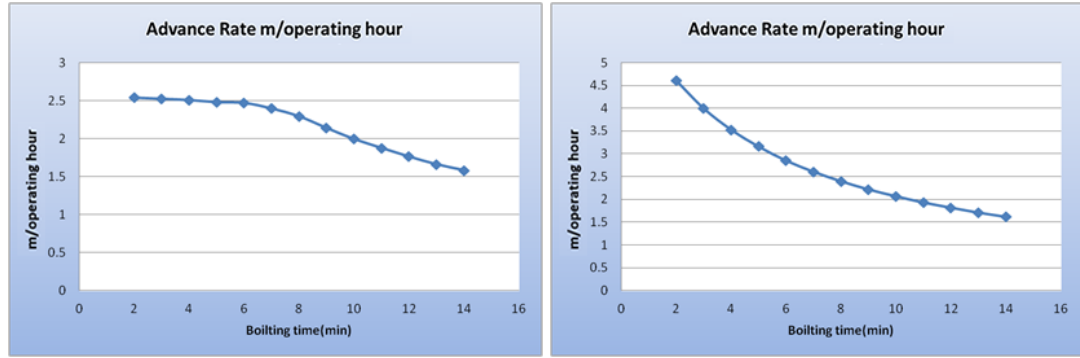


(a) Single SC

(b) CHS

Figure 6-68 Comparison of SC against CHS Mine 2

The same experiment was done for the Mine 3 scenario (Figure 6-69), but as the figure shows, the development performance only doubled to 4.6 m/operating hour with an extremely fast support cycle because Mine 3 used MBs, which must stop repeatedly for support operations in each cycle.



(a) One SC for each CM

(b) CHS

Figure 6-69 Comparison of SC against CHS of Mine 3

6.8 Analysis of constraints - support constrained or transport constrained

In a roadway development environment a number of cyclic processes and a range of equipment exists and in a given scenario, any of these processes or equipment can be a potential constraint. A process might appear to be support constrained at the start of the pillar cycle but transport constrained at the end when the distance from the boot end has increased above a certain threshold.

One of the most suitable techniques for determining whether a process is support or transport constrained is to plot the process rates (metres advanced/operating hour) against the increasing distance (metres) from the boot end (Porteous, 2008). Based on the slope of the resulting graph, conclusions about the nature of the constraint can be made. For example, consider the following graph (Figure 6-70):

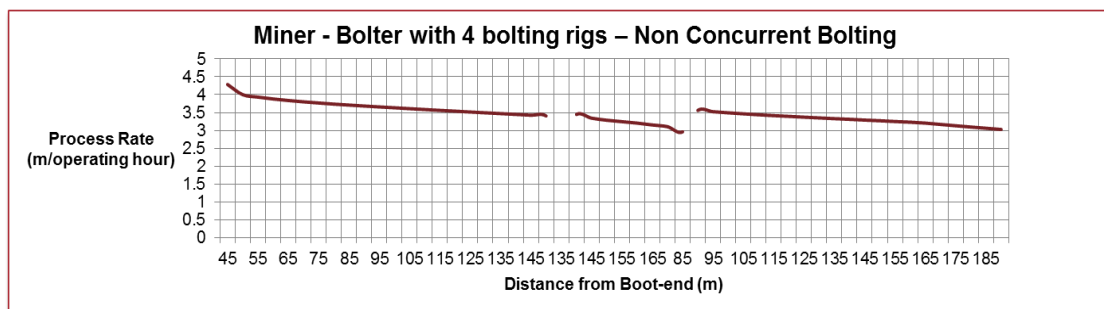


Figure 6-70 Process rate of non-concurrent bolting

As the graph shows, the process rate in this scenario remains more or less constant irrespective of the distance from the boot end, but as the distance from the boot end increases, the demand placed on the transport system increases as well. Now, if the capacity of a transport system is equal to or greater than the demand placed on it, the process rate tends to remain constant despite the increasing distance from the boot end. In such a case, it is reasonable to deduce that the system is support constrained.

Now, consider another scenario illustrated by the graph below (Figure 6-71):

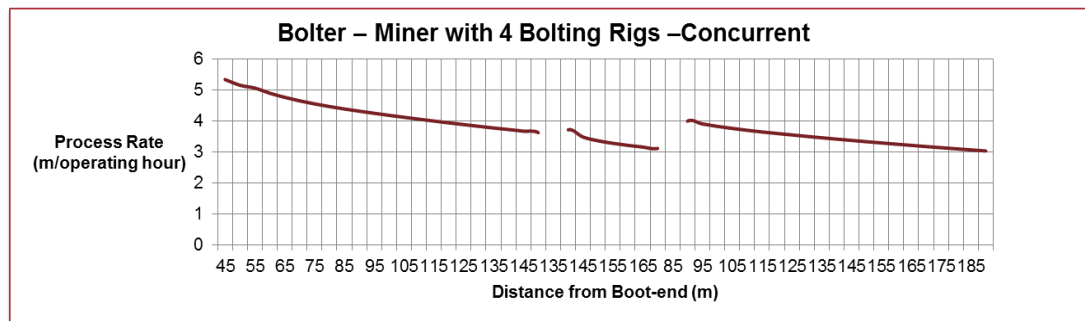


Figure 6-71 Process rate of concurrent bolting

In this case the process rate tends to drop as the continuous miner moves away from the boot end so it is reasonable to deduce that the system is transport constrained because here the capacity of the transport system was less than the demand placed on it.

In general, if the process rate drops considerably with increasing distance from the boot end, then the system is likely to be transport constrained but if the process rate remains constant then the system is most likely to be support constrained (Porteous, 2008). However, much more complex scenarios than the one described above may be encountered, such as where the process rate may remain constant for a considerable distance and then begin to drop. In such ambiguous cases, a delay state analysis combined with the above methodology can be used to identify the nature of the

constraint. The roadway development model was used to demonstrate how a process rate analysis can be combined with a delay state analysis for this purpose.

6.8.1 The model configuration

Using the Flexsim model, a series of two-heading roadway development scenarios were simulated to demonstrate how a combination of delay state and process rate analysis can be used to continually improve the efficiency of the pillar development cycle:

- Scenario 1: miner bolter (MB) continuous miner (CM) configuration with only one bolting rig on either side, i.e., a total of only two bolting rigs;
- Scenario 2: MB CM configuration with two bolting rigs on either side of the CM, i.e., a total of four bolting rigs;
- Scenario 3: bolter miner (BM) CM configuration with two bolting rigs on each side of the CM, i.e., a total of four bolting rigs;
- Scenario 4: BM CM configuration with two bolting rigs on each side of the CM, i.e., a total of four bolting rigs along with self-drilling bolts (SDBs);
- Scenario 5: BM CM configuration with two bolting rigs on each side of the CM, i.e., a total of four bolting rigs and two shuttle cars (SCs);
- Scenario 6: BM CM configuration with two bolting rigs on each side of the CM, i.e., a total of four bolting rigs and two SCs along with self-drilling bolts (SDBs);
- Scenario 7: BM CM configuration with two bolting rigs on each side of the CM, i.e., a total of four bolting rigs and a continuous haulage system (CHS);
- Scenario 8: BM CM configuration with two bolting rigs on each side of the CM, i.e., a total of four bolting rigs with a CHS and SDBs;

The main model configuration parameters used for the various scenarios are listed below:

- roadway dimensions: $W = 5$ m, $H = 3$ m;
- pillar dimensions: $L = 110$ m, $W = 40$ m;
- boot end to cut through = 20 m;
- over drive length = 20 m;
- length of gas drainage stub = 5 m;
- overdrive = 20 m;
- support density: six roof bolts and four rib bolts per metre advance;
- average time per web = 1.5 minutes;
- average time before cutting = 0.33 minutes;
- average time after cutting = 0.5 minutes;
- average time of normal bolts = 3 minutes;
- average time of self-drilling bolts = 1.88 minutes;
- average time to relocate the miner to a different heading = 20 minutes;
- average time of panel advance delay = 22 hours.

The simulation times were recorded automatically for every five metres advance. The time taken to develop five pillars improved from 568 operating hours in scenario 1 to 293 operating hours in scenario 8 (Table 6-10).

Table 6-10 Summary of the scenarios and the cycle times to complete 5 pillars

Scenario	Miner Type	Number of Bolting Rigs	Type of Bolts	Haulage Type	Total Hours for 5 Pillars
1	Miner Bolter	2	normal bolts	1 SC	568
2	Miner Bolter	4	normal bolts	1 SC	490

3	Bolter Miner	4	normal bolts	1 SC	461
4	Bolter Miner	4	SDBs	1 SC	461
5	Bolter Miner	4	normal bolts	2 SCs	410
6	Bolter Miner	4	SDBs	2 SCs	405
7	Bolter Miner	4	normal bolts	CHS	366
8	Bolter Miner	4	SDBs	CHS	293

6.8.2 Analysis of the simulation results

Figure 6-72 to Figure 6-89 illustrates the results obtained from simulating the above scenarios. The graphs included in this section indicate the variations in the process rate over a single pillar development cycle, so they did not include the panel advance delay. In the first scenario, a miner-bolter CM configuration with two bolting rigs and one shuttle car was simulated. The results of the simulation are as shown in Figure 6-72.

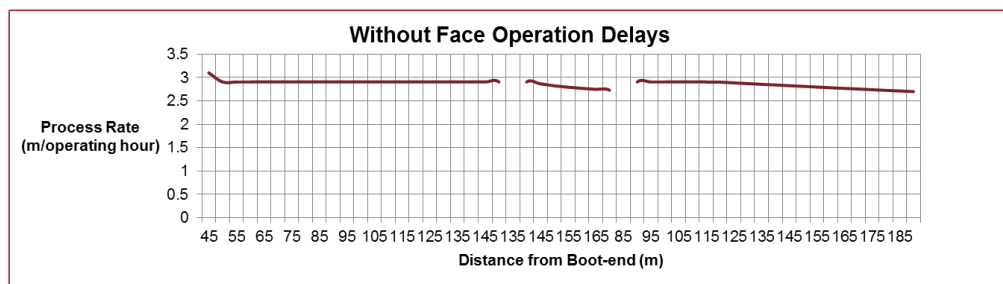


Figure 6-72 Simulation results for scenario 1

The slope of the graph indicates that the process rate did not vary much as the distance from the boot end increased, indicating that the process was support constrained.

The above scenario was simulated without taking the face operation delays into account and did not include delays due to activities such as ventilation tube extension, stone dusting and supplying the continuous miner. Figure 6-73 shows the process rate graph

when delays such as these were considered. This graph also shows that these ancillary operations or the parallel processes tended to have a negative effect on the process rate, whereas a linear trend line plotted through the graph indicated that the process rates did not vary much as the distance from the boot end increased.

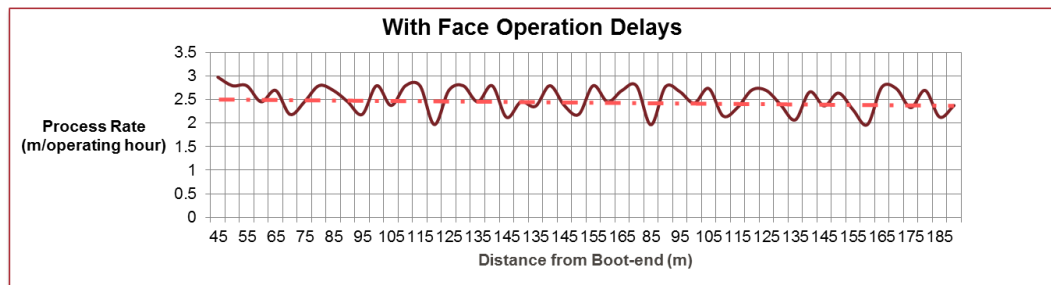


Figure 6-73 Simulation results for scenario 1 including the face operation delays

For subsequent scenarios the results were presented without taking delays of these face operations into account to ensure clarity of illustration for the purposes of this study as they would add unnecessary fluctuation to the results. However, in practical scenarios such delays would have a considerable impact and as a consequence the actual process rate may be quite different from the simulation results shown in Figure 6-73.

Figure 6-74 is a pie chart generated by Flexsim[®], which illustrates the states profile of the development process and shows that supporting was the main constraint and it took up 61% of the total calendar time.

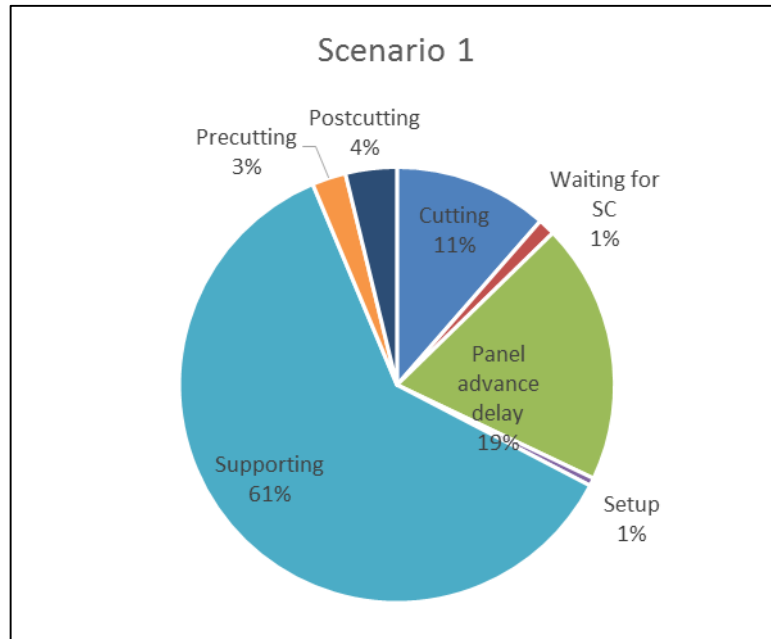


Figure 6-74 Delay state analysis for scenario 1

The introduction of additional bolting rigs in scenario 2 was expected to improve the process rate compared to the original scenario. The results of this simulation are as shown in Figure 6-75, with the graph showing that introducing additional bolting rigs improved the process rates. Indeed the system was in a transition state and was moving towards being transport constrained. However, variations in the process rates with distance from the boot end was only minor, while the delay state chart (Figure 6-76) suggests that support activities were still the biggest source of delays and indicates that the process was still support constrained.

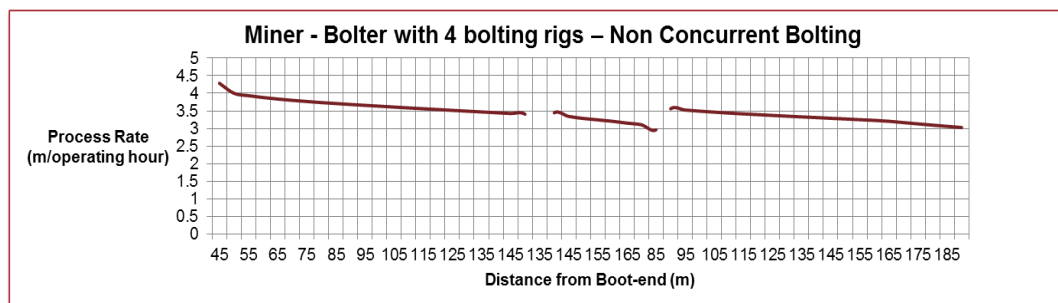


Figure 6-75 Simulation results for scenario 2

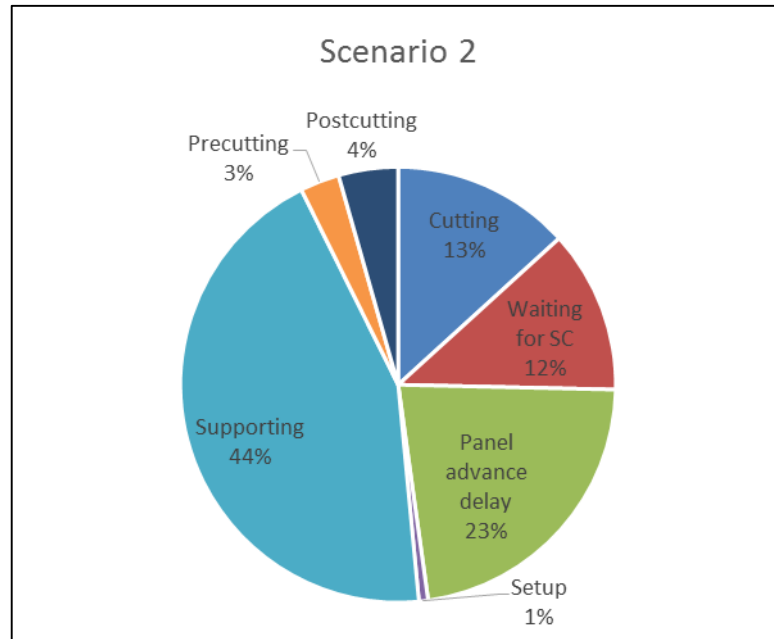


Figure 6-76 Delay state analysis for scenario 2

In a number of practical scenarios, the process rate analysis might not be conclusive enough to determine the nature of constraints, such as with scenario 2 for example, where the process rates varied to a certain extent making it difficult to ascertain whether the process was in a transition state or had become transport constrained. In such complex scenarios, the process rate analysis can be combined with a delay state analysis to yield conclusive results. The delay charts included in this section depict the delay state of the system over a five pillar development cycle and have been generated using the Flexsim simulation model.

Figure 6-77 illustrates the results from the simulation of scenario 3. The slope of the graph suggests a shift in the nature of the constraints compared to scenario 2. A bolter-miner allowed the cutting and bolting operations to be performed in parallel which improved the performance considerably and resulted in a shift in the nature of the constraint. The delay state chart (Figure 6-78) confirmed this shift because the transport

system was then the biggest source of delay. Clearly, the system was then transport constrained.

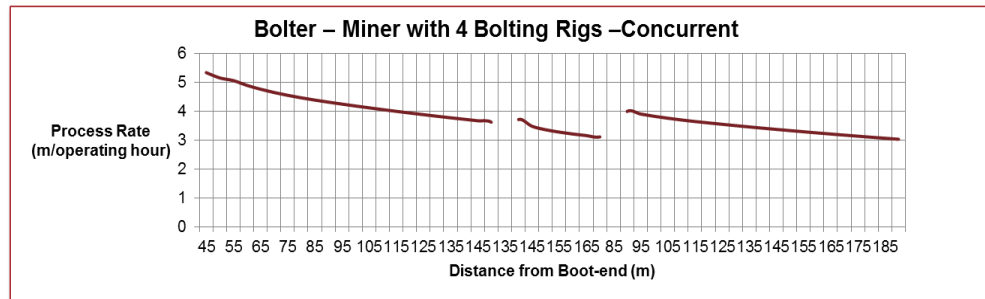


Figure 6-77 Simulation results for scenario 3

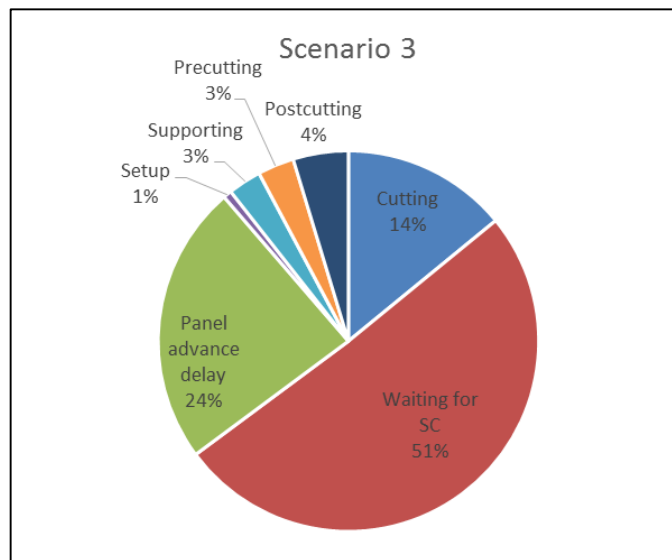


Figure 6-78 Delay state analysis for scenario 3

The reason for simulating scenario 4 was to prove that an improvement in the productivity of a non-bottleneck process would not lead to an improvement in system performance. As discussed before, the roadway development configuration represented by scenario 3 was transport constrained, so an improvement in the support mechanisms should not lead to an improvement in system performance. In scenario 4, self-drilling bolts in replace of conventional bolts were used while keeping all other parameters the same as in scenario 3. The results from the simulation are depicted in Figure 6-79. The

graph shows that the overall process rate remained unchanged, while the delay state chart shown in Figure 6-80 confirmed this view while indicating that the process was transport constrained. The time taken to develop five pillars as determined from the simulation model was 461 operating hours, which is exactly the same as scenario 3.

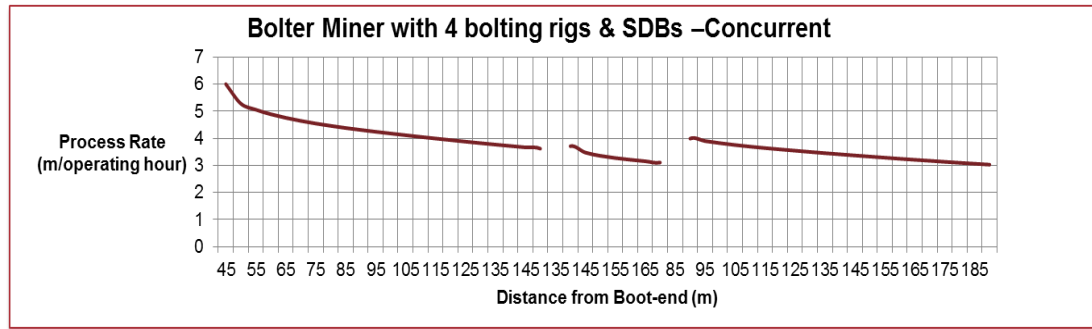


Figure 6-79 Simulation results for scenario 4

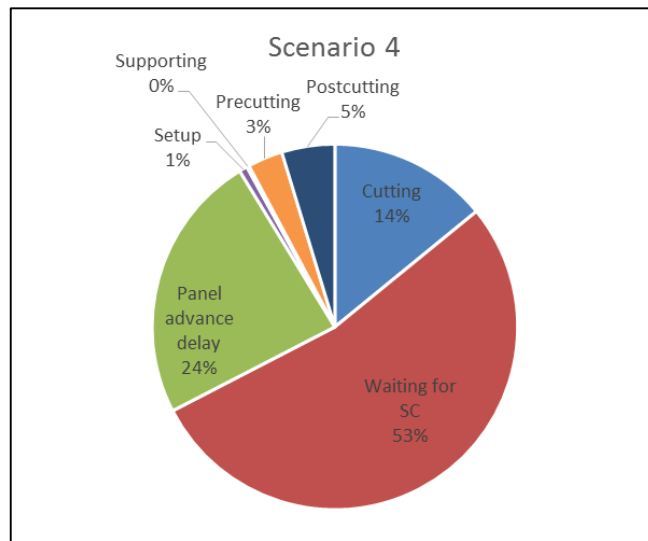


Figure 6-80 Delay state analysis for scenario 4

In scenario 5 a second shuttle car was introduced to keep all the other parameters the same as scenario 3. An improvement in capability of the transport system further improved the process rate, as illustrated by the results from the simulation shown in Figure 6-81. It must be noted here that although the productivity of the system was improved, the nature of the constraint was still not changed. This scenario is another

good example of complex cases where a combination of process rate analysis and delay state assessment is required to ascertain the true nature of the constraint. In this case, by using the process rate analysis alone, it may be difficult to determine the nature of a constraint but the delay state chart (Figure 6-82) shows that “waiting for SC” is still the biggest source of delay and also indicates that the process is still transport constrained.

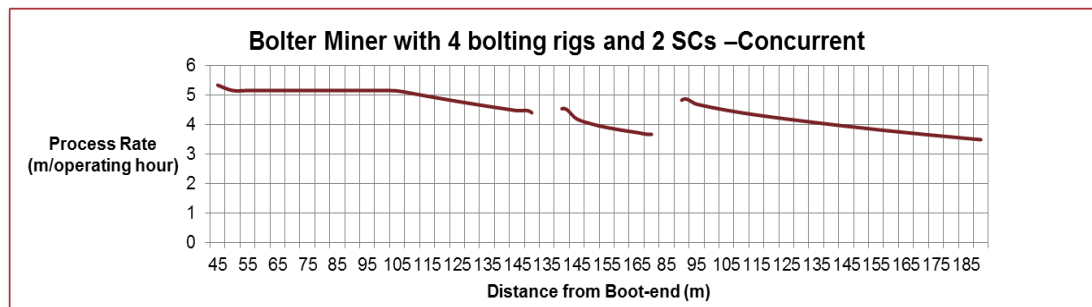


Figure 6-81 Simulation results for scenario 5

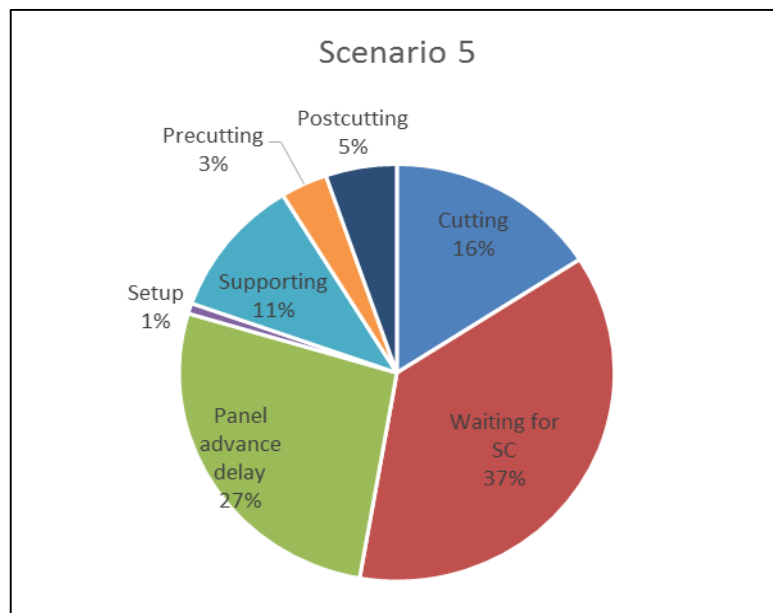


Figure 6-82 Delay state analysis for scenario 5

In scenario 6, self-drilling bolts were used while keeping all the other parameters the same as in scenario 5. Figure 6-83 shows that although the process rates were higher at the start of the cycle, the overall process rate remained unchanged compared to scenario 5 because the system was transport constrained (Figure 6-84) and therefore an

improvement in support capability did not improve system productivity. The time taken to develop five pillars as determined from the simulation model was 405 operating hours, which is a marginal improvement over scenario 5 (410 operating hours) but not substantial enough to justify an investment in self-drilling bolts.

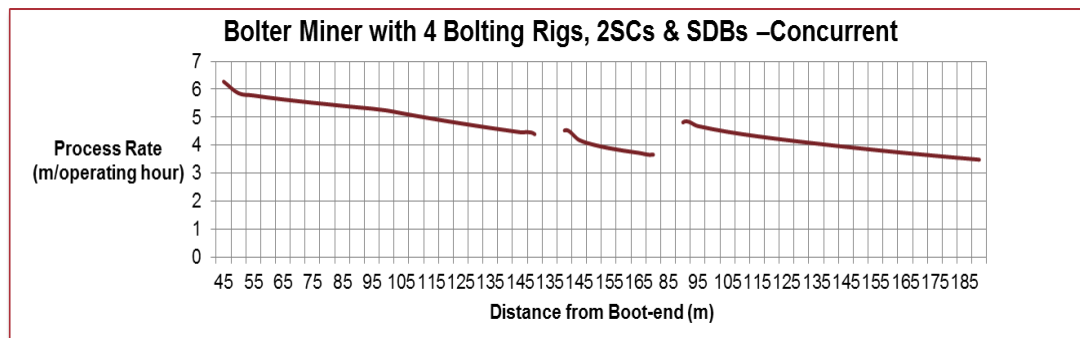


Figure 6-83 Simulation results for scenario 6

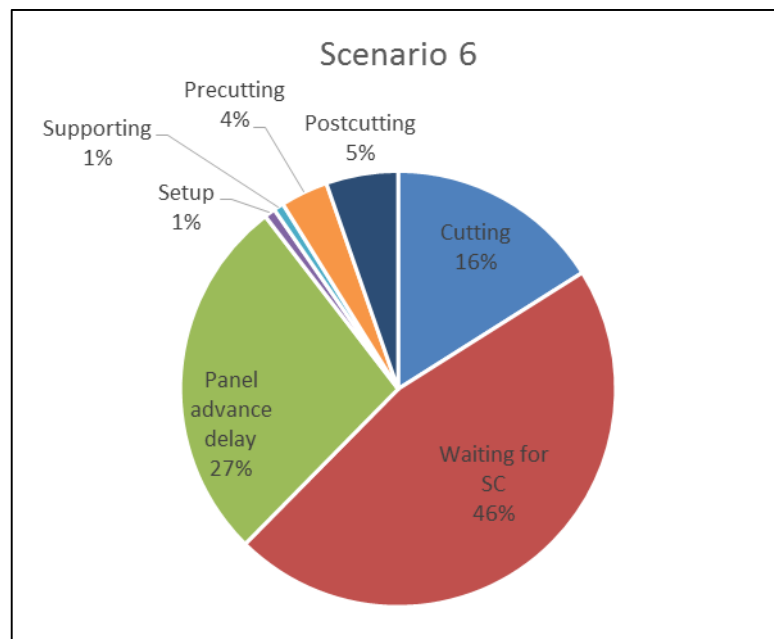


Figure 6-84 Delay state analysis for scenario 6

The introduction of a continuous haulage system (CHS) in scenario 7, however, produced some considerable improvement. The system process rate improved considerably, as shown in Figure 6-85 and the system was now support constrained. This was expected because a CHS has a much higher transport capacity than a shuttle

car. The time taken to develop five pillars as determined from the simulation model was 366 operating hours, which was a considerable improvement over scenario 6.

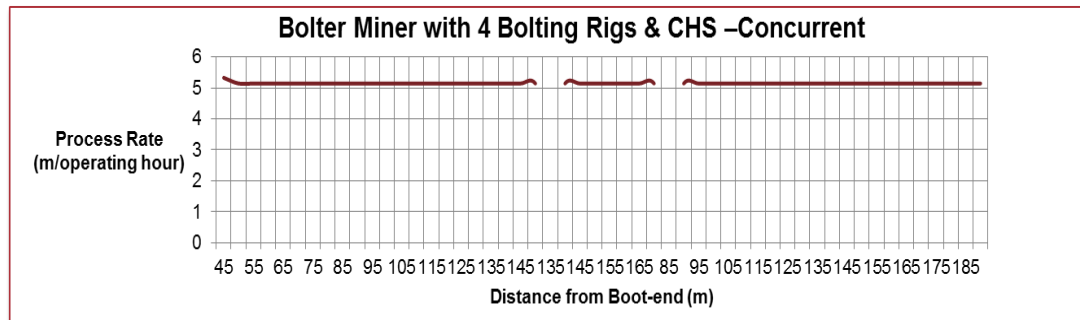


Figure 6-85 Simulation results for scenario 7

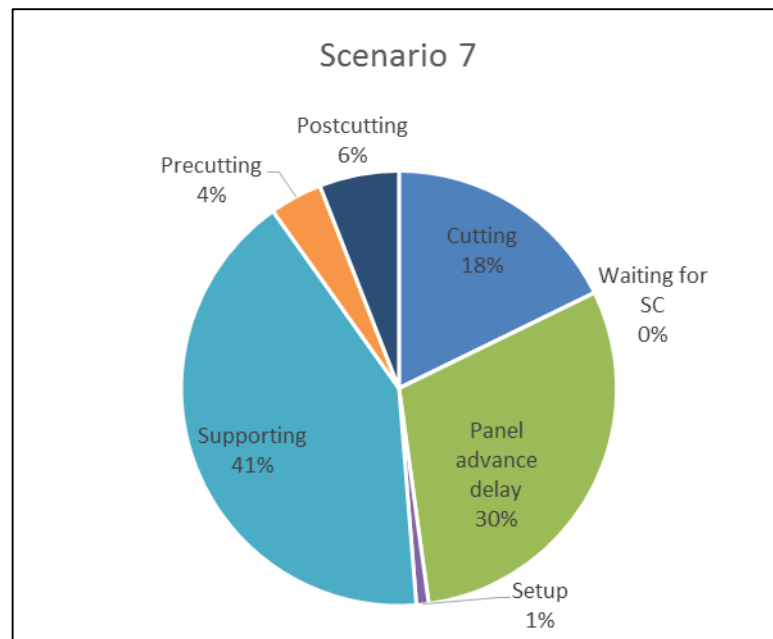


Figure 6-86 Delay state analysis for scenario 7

The introduction of self-drilling bolts instead of normal bolts in scenario 8 improved the process rates, as shown in Figure 6-87 and Figure 6-88.

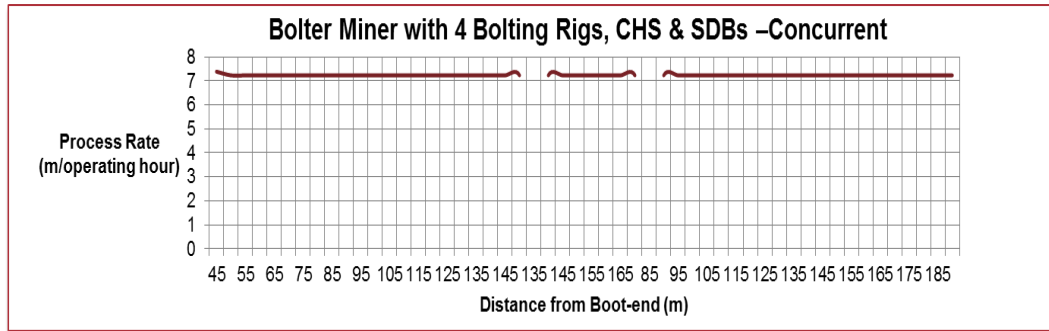


Figure 6-87 Simulation results for scenario 8

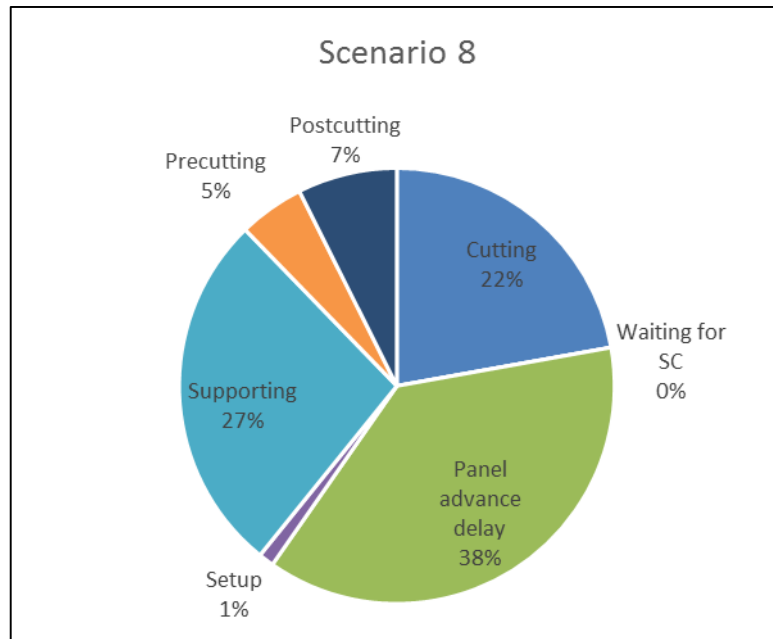


Figure 6-88 Delay state analysis for scenario 8

As illustrated above, once a productivity improvement measure has been implemented, the system performance must be re-assessed to identify changes in the nature of the constraint and then the whole cycle of improvement must be repeated again. The time taken to develop five pillars improved from 568 operating hours in scenario 1 to 293 operating hours in scenario 8. However, the on-going improvement process does not stop here. From the delay state charts (Figure 6-86 and Figure 6-88) of scenario 7-8, it is evident that a major delay was initiated by the panel advance processes.

The simulation results for the eight scenarios are summarised in Figure 6-89.

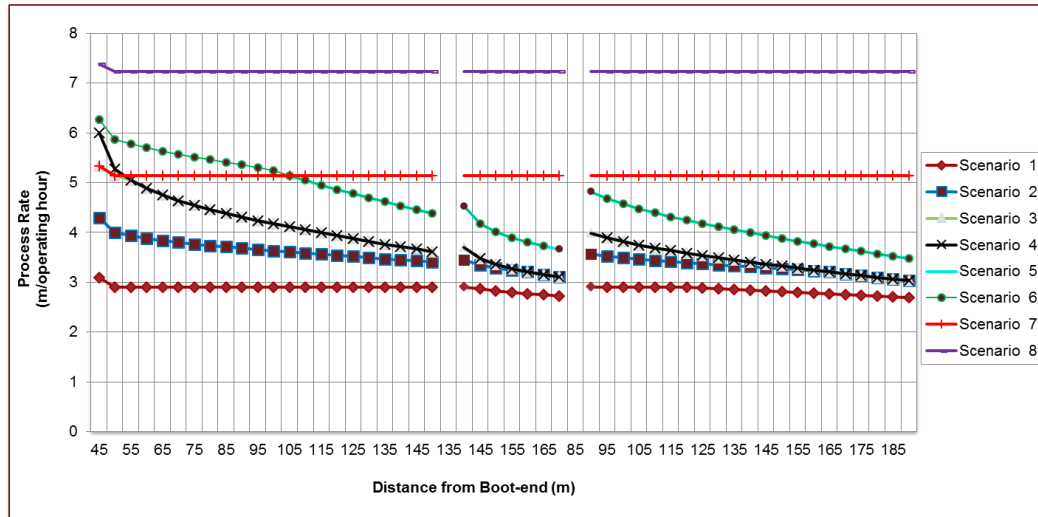


Figure 6-89 Simulation results for the eight scenarios

6.9 Summary of the roadway development model

A general purpose roadway development model has been designed. Because of its configurable and structured design, the model can evaluate alternative “what if” scenarios and reproduce statistically varied random and non-random events and analysis over a period of time. This makes it possible to assess how a particular configuration may perform with high levels of variability and uncertainty in operations.

The case studies demonstrated a great use for the developed model in the way of single impact, multi impacts and an analysis of the constraints. As the case studies show, because roadway development is a stop start process, the main bottlenecks are either the support cycle or the SC tramming cycle. The support cycle tends to have more impact on the development rate than the continuous miner cutting cycle, so for longer support cycles, the SC’s speed and its dumping time had a limited effect on the development rate. For an existing system, only enhancing one or two processes would mean little improvement to the overall performance; improvements should be made to a set of processes or the bottleneck of the system. Utilisation of a CHS can significantly

improve the development rates, but any improvement is restricted by the bolting time. Also, the impact of a CHS in combination with a BM is much more than in combination with a MB.

7 CONCLUSION AND RECOMMENDATIONS FOR FUTURE RESEARCH

7.1 Summary

This thesis described the modelling, simulation and analysis of longwall mining systems using Flexsim[®]. The study successfully modelled the longwall production face and the roadway development process using discrete event method and studied the longwall mining delays, production, technologies and strategy optimisation using the developed models. The research offered critical insights into mining performance by analysing mine data, systems modelling and sensitivity analyses.

Original delay and production data were sourced from Australian longwall coal mines. Historical data collected from Australian mines was analysed to highlight the features for modelling perspective and data for model design, input and validation was categorised.

In one of the longwall faces only around 30% of total calendar time was utilised in production activities and in the roadway development units analysed, a large amount of time was lost due to delays and other non-production activities. Therefore, the delay time had to be considered when modelling the system and the production processes.

The delay data had to be processed before fitting appropriate statistical distributions to the data. Processing included combining continuous and overlapping delays and removing long and abnormal delays, outliers which would skew the statistical fit, for special consideration.

Two models of longwall mining systems were developed and used to evaluate various potential strategies for increasing production. The models can either be configured using data from an existing longwall to highlight any bottlenecks, or with projected data for a newly designed longwall system in order to evaluate its potential performance and make

changes before committing to the purchase of expensive equipment for the actual mine. The models allow for many replications of multiple scenarios to be run with multiple sets of variables, while integrating a full range of mining processes and delays, with randomness and variability, into a single and best practice simulation platform. It means that an entire process can be analysed and improved in context.

The case studies showed that the validated models can evaluate potential alternative ways to improve longwall production and roadway development, while reproducing statistically varied random and non-random events over a period of time.

A sensitivity analysis was made using a calibrated longwall model to examine how a range of AFC capacities, shearer speeds, support operation times, face widths and web depths would impact on production. The results demonstrated the impact that individual machine parameters and the design of the longwall would have on production, highlighted the constraints and offered potential alternatives to improving production.

The simulation of the roadway development process revealed that in the stop-start nature of the roadway development process the support cycle or the SC tramming cycle are the main bottlenecks. The support cycle tended to have more impact on the development rate than the CM cutting cycle, whereas the SC's speed and unloading time had a limited effect on the development rate when the support cycle is long. With an existing system, enhancing only one or two processes makes almost no contribution to the overall performance; improvements should be made to a set of strategies or the bottleneck of the system. Utilisation of a CHS can significantly improve development rates, but any improvement is restricted by the bolting time and the improvement obtained is much greater in combination with a BM than with a MB.

7.2 Methodological contribution

The major outcomes of this study were the development of general purpose simulation models of the longwall production face and the roadway development process. The research incorporated the complicated systems, different types of machines and potential alternatives into discrete event simulation models. This benefits the mining industry by demonstrating a new philosophy for evaluating the performance of longwall mining and roadway development cycles in underground coal mines. While it takes longer to develop a general purpose model compared to models that only work for one or a few scenarios, the additional effort resulted in a more detailed analysis and the work of potential model users and analysts will become easier. This study provides the means for a diversity of perspectives and new ideas for thinking about changes to be examined and offers methods for assessing the uncertainty of the performance of longwall production systems and roadway development processes. These include:

- i. a framework to enable a longwall mining face and roadway development to be modelled in a structured way;
- ii. an innovative way to model the longwall mining process with configurable cutting sequences;
- iii. an innovative way to model the interactions among supports, the shearer and the AFC, which allows the size of the longwall face to be easily changed;
- iv. an innovative way to model the roadway development process with configurable development sequences and allow many possible roadway layouts, including multi-headings, to be simulated;

- v. an innovative way to model the face operational delays and the support cycle; the model is flexible enough to be configured with as many face operation delays and support types as the user may encounter.

7.3 The features of the models

The models have the following features:

- i. the models are oriented with 3D objects. Each object can work as a real piece of longwall face or roadway development equipment and work together as a longwall production system. The models perform in a virtual reality environment. What can be seen from the model is what would actually happen in a longwall mining face and a roadway development unit. 3D views and animation of the model allow users to view the dynamic production system as it operates, resulting in a far better understanding of the actual operation;
- ii. the models incorporate real-world interactions and delays such as breakdowns, shift schedules and interactions between machines;
- iii. the configurable and structured model design with optional strategies and user friendly GUIs makes the model easy to use. There is no need to have advanced knowledge of Flexsim® to use this model, making it a useful tool for those without a programming background, while still having ultimate flexibility for advanced users;
- iv. the longwall model can simulate most longwall layouts by changing values such as the width and length. It can also simulate most cutting sequences by configuring a numerical table;
- v. the roadway development model can simulate many standard roadway layouts by simply changing the values such as the width and length, while having the

flexibility to simulate a gateroad or mainway of multi headings using the task sequence table. The model can also simulate different miner types and complicated development sequences by configuring a numerical table, while retaining the flexibility to configure any face operational delay. The design of the support cycle makes it possible to configure any standard support type and multi support types can be applied in different sections in one pillar. Multi haulage strategies such as one SC, two SCs or CHS and real time SC interactions can also be represented;

- vi. the ability to evaluate alternative “what if” scenarios, reproduce statistically varied random and non-random events and analyses over a period of time. Using Flexsim’s integrated analysis tool makes it easy to perform multi-scenario experiments and an entire development process can be easily analysed and improved in context. The ability to evaluate an existing system or potential longwall mining technologies and practices, integrating emerging technologies and subsystems, prior to costly field trials, can lead to a rapid adoption of new technology while minimising uncertainty and risk. It offers high quality information and identifies high quality solutions rapidly, which is a key to the success of any ongoing project.

7.4 Limitations

The longwall model and the roadway development model work separately, but ideally they should be modelled as one interdependent system.

As discussed, the longwall model has the following issues:

- the time units must be in seconds because the operation of one support unit is in seconds. The model needs to be modified so that the time unit for any piece

of equipment can be pre-defined and therefore act seamlessly with other equipment;

- the run speed of the model is limited, ideally not greater than 500 simulation seconds per real time second, which means, for example that, if the model is required to simulate production for 500 hours, it would take at least 1 hour to obtain the results. If the model runs too fast, the time interval between location refreshments is large in real time, the shearer may jump to a new location without waiting for the support operation and the model would then stop and never resume;
- the user must manually convert the conveyor capacity of tonnes/hour to conveyor speed of metres/second, before configuring the model.

The issues of the roadway development model are as follows:

- the design of the roadways must be in repetitive blocks. The model only repeats one pillar of development and all the configurations such as the pillar size and the development task sequence are based on one pillar, apart from the downtime which may be collected over a long period of time;
- the model cannot simulate the place change method, it can only simulate in-place methods by a CM with mounted rigs.

7.5 Recommendations for Future Research

This thesis is on integrating discrete event simulation modelling into longwall mining including the development process using Flexsim®. Apart from the limitations identified above, which need further effort to overcome, some other work is required to further the integration of discrete event simulation into longwall mining systems.

First, more data should be obtained from mine sites. For this research, only one mine's longwall data for a one year period and two mines' roadway development units were collated and analysed. As discussed, data recorded by machine data loggers and reported by shifts are not sufficient for a realistic model. Some of the data, such as the longwall shearer speeds at different stages, support advance time, CM cutting and loading time, the SC travel speed and discharge time, are not recorded by any data logger. This kind of data is hard to collect unless additional data loggers are installed in key areas.

Second, the models developed in this study can only work with a licensed version of Flexsim[®] and there are several shortcomings:

- anyone who wants to use the models and perform some analysis must initially have access to a licensed Flexsim[®] or buy a Flexsim licence, which is expensive. Also the compatibility among different Flexsim versions is not satisfactory and some unexpected errors may occur when using a different version from the version in which the models were created;
- it is difficult to change some of the basic functions because Flexsim[®] is a closed source commercial software. It is a general purpose simulation software, it was not designed specifically for mining. Mining equipment often has special features. For example, a conveyor that dynamically changes its length during simulation; there is no way to get access to the background codes and change the properties of the *Conveyor* object of Flexsim[®], unless the developer of Flexsim[®] updates it upon request. Flexsim[®] does offer the feature to use external .dll files or write customised codes using C++. It would probably take about the equivalent amount of work to develop such features for all mining

equipment in Flexsim[®] as it would take to develop a new simulation tool using an open source code;

- there are many functions and terminologies in Flexsim[®] that mining engineers may not be familiar with. Although Flexsim[®] is a good tool and relatively easy to learn, it would still be hard work for a mining engineer who knows nothing about simulation or Flexsim[®], to perform an analysis using the developed models. For example, if the user wanted to perform a multi-scenarios experiment, it is necessary to know what a tree structure is and where a variable is stored.

By using an open source discrete event simulation software and developing a standalone longwall simulation tool, there would be no such problems. Although there are always some difficulties and risks when developing a new program, the benefits would surpass these. Open source codes have more flexibility for designing a tool for mining simulation from the very beginning. Every imaginable function can be designed without worrying about constraints. Constant development can be carried on from basic functions and expanded in future updates.

REFERENCES

- Abu-Taieh, E. M. O. and EL Sheikh, A. A. R. 2007. Commercial simulation packages: a comparative study. *International Journal of Simulation Systems, Science and Technology, Special Issue on: Models and Applications in Engineering and Technology*, 8, pp 66-76.
- Adenso-Díaz, B., Lev, B. and Artime, R. 2004. Simulation in dynamic environments: optimization of transport inside a coal mine. *IIE Transactions*, 36, 547-555.
- Askari-Nasab, H., Frimpong, S. and Szymanski, J. 2007. Modelling open pit dynamics using discrete simulation. *International Journal of Mining, Reclamation and Environment*, Volume 21, Issue 1, pp 35-49.
- Baafi, E., Gray, G., Porter, I. and Rojas, O. 2009. The modelling of roadway development to support longwall mining. Brisbane, Queensland, Australia: Coal Association Research Program (ACARP).
- Baafi, E. and Porter, I. 2012. Underground coal mine delay data analysis system. In: *Mine Planning and Equipment Selection*. Irvine, California, USA. pp 60-68.
- Barton, R. R. 2003. Panel simulation: past present and future. In: *Proceedings of the 2003 Winter Simulation Conference*, New Orleans, Louisiana, USA. pp 2044-2050.
- Basu, A. J. and Baafi, E. Y. 1999. Discrete event simulation of mining systems: current practice in Australia. *International Journal of Mining, Reclamation and Environment*, 13, pp 79-84.
- Bauer, A. and Calder, P. N. 1973. Planning open pit Mining operations using simulation. In: *Proceedings of APCOM*, Johannesburg, South Africa. South African Inst. of Mining and Met., pp 273 - 278.

- Beaverstock, M., Greenwood, A. G., Lavery, E., Nordgren, B. and Warr, S. 2011. *Applied simulation: modelling and analysis using Flexsim*, Flexsim Software Products Incorporated.
- Bise, C. J. and Albert, E. K. 1984. Coal - Comparison of model and simulation techniques for production analysis in underground coal mines in the book: AIME Transactions Volume 276.
- Birchall, G. 2007. 703 MG Block Management Plan of Appin, Illawarra Coal, Personal communication with Gary Gibson, 2011.
- Bosilj-Vuksic, V., Ceric, V. and Hlupic, V. 2007. Criteria for the evaluation of business process simulation tools. *Interdisciplinary Journal of Information, Knowledge, and Management*, 2, pp 73-88.
- Bucklen, E. 1968. *Computer applications in underground mining systems*, Dept. of Engineering, Virginia Polytechnic Institute and State University.
- Bucklen, E. P., Subuleski, S. C., Prelaz, L. J., Lucas, J. R., 1969. Computer applications in underground mining systems. U. S. Dept. of Interior Res. & Dev. Report #37.
- Cross B. and Williamson, G., 1969. Digital simulation of an open-pit truck haulage system. In: Proceedings of the APCOM 1969, Soc. of Mining Engrs., Denver, CO, USA.
- David A., Dennis S., Thomas W., Jeffrey C. and James C. 2012. In Chapter 16 - Simulation, Quantitative Methods for Business. Published by Cengage Learning.
- Dornetto, L. 1988. An adaptive control scheme-expert system-that optimizes the operation of a proposed underground coal mining system with applications to short wall, longwall and room and pillar mining systems. *In: Systems, Man, and*

- Cybernetics. Proceedings of the 1988 IEEE International Conference on, 1988. IEEE, pp 209-214.
- Expertfit website, 2013. [Online]. Available: <http://www.averilllaw.com/distribution-fitting/> [Accessed 2013].
- Flexsim user manual, 2013. [Online]. Available: <http://www.flexsim.com/community/forum/downloads.php?do=file&id=116> [Accessed 23.6.2013].
- Flexsim website, 2013. [Online]. Available: <http://www.flexsim.com/company/> [Accessed 23.6.2013].
- Flexsim website, 2014, Mining Simulation, Available: <https://www.flexsim.com/mining-simulation/>, [Accessed 11.12. 2014].
- Geoscience, A. and Abare, 2011. Australian Bureau of Agricultural and Resource Economics-Australian Resource Assessment. *Chapter 5 Coal*.
- Gibson, G. 2005. Australian roadway development – Current practices. Brisbane, Queensland, Australia: Coal Association Research Program (ACARP).
- Gibson, G. 2013. ACARP Roadway Development Improvement Project. Presentation on 2013 Longwall Conference. 14-15 October, 2013, Hunter Valley, NSW, Australia.
- Greberg, J. 2011. Simulation as a Tool for Mine Planning. In: Proceedings of the Second International Future Mining Conference. November 22-23, Sydney, NSW, Australia.
- Hall, R. A. 2000. Reliability analysis and discrete event simulation as tools for mining equipment management. Available:

http://www.collectionscanada.gc.ca/obj/s4/f2/dsk1/tape4/PQDD_0035/NQ52824.pdf; [Accessed 20.3.2012]

- Hancock, W. S. and Lyons, D., K. G., 1987. Operational research in the planning of underground transport. In: Proceedings of the APCOM 18, London, UK.
- Haycocks, C., Kumar, A., Unal, A. and Osei-Agyemana, S. 1984. Interactive simulation for room and pillar miners. *In: Proceedings of 18th APCOM, Institute of Mining and Metallurgy, London, England*, pp 5591-5597.
- Holla, L., Barclay, E. 2000. Mine Subsidence in the southern coalfield, NSW. Department of Mineral Resources, NSW, Australia.
- Jenkins, C. M. and Rice, S. V. 2009. Resource modelling in discrete: event simulation environments: a fifty-year perspective. *In: Proceedings of Winter Simulation Conference, Austin, TX, USA*, pp 755-766.
- JOY Gloable [Online]. Available: <http://www.joy.com/Joy/Products.htm> [Accessed 20.5.2013].
- Juckett, J. E., 1969. A coal mine belt system design simulation. In: Proceedings of the 3rd Conference on the Application of Simulation, Los Angeles, CA, USA.
- Kreutzer, W (1986), Systems Simulations – Programming Styles and Languages, Addison-Wesley.
- Koenigsburg, E. 1958. Cyclic queues. *Operations Research*, 22-34.
- Kolonja, B., Kalasky, D. R. and Mutmanský, J. M. 1993. Optimization of dispatching criteria for open-pit truck haulage system design using multiple comparisons with the best and common random numbers. *In: Proceedings of the Winter Simulation Conference, Los Angeles, CA, USA*, pp 393-401.

- Lavery, E. 2008. Introduction to Flexsim DS. 2013. [Online]. Available: <http://www.flexsim.com/> [Accessed 2013].
- Madge, D. N., 1964. Simulation of truck movement in an open pit mining operation, Can. Oper. Res. Soc., pp 32 - 41
- Manula, C. and Rivell, R., 1974. A master design simulator. In: Proceedings of the 12th APCOM, Col. School of Mines, Golden, CO, USA
- Manula, C. and Venkataraman, N., 1967. Open pit haulage simulation, Internal Dept. Rept. the Pennsylvania State University, College Station, PA, USA.
- Mcneary, R. L. and Nie, Z. 2000. Simulation of a conveyor belt network at an underground coal mine. *Mineral Resources Engineering*, 9, pp 343-355.
- Mitchell, G. W. 2009. Longwall Mining. In: *Australian coal mining practice chapter 5*. 3rd Edition ed. Kininmonth, R. J. and Baafi, E. Y. (Eds.). Australian Institute of Mining and Metallurgy. Carlton, Vic, Australia.
- Moignard R., 3D Experience for Mining Productivity & Profitability. <http://www.3ds.com/fileadmin/COMPANY/Investors/Presentations/Feb19-2013-3DEXPERIENCE-for-Mining.pdf>, [Accessed 12.11.2014].
- Nance, R. E. 1993. A history of discrete event simulation programming languages. In: *Proceeding HOPL-II The second ACM SIGPLAN conference on History of programming languages*, New York, NY, USA, pp 149-175
- Nance, R. E. 1996. A history of discrete event simulation programming languages. In: *History of programming languages - II*. ACM, pp 369-427.
- Newhart, D. D., Bethbelt-1, 1977. A belt haulage simulator for coal mine planning. Research Report File 1720-2, Bethlehem Steel Corporation, Research Department, Bethlehem, PA, USA.

- Newman, A. M., Rubio, E., Caro, R., Weintraub, A. and Eureka, K. 2010. A review of operations research in mine planning. *Interfaces*, 40, pp 222-245.
- Nienhaus K, Bayer A K and Haut H. Higher Productivity - A Question of Shearer Loader Cutting Sequences, Aachen University of Technology, Germany. <http://www.bgmr.rwth-aachen.de/downloads/hpmpartikel.pdf> ; Retrieved on August, 2, 2012.
- Nordgren, W. B. 2003. Flexsim simulation environment. *In: Proceedings of Winter Simulation Conference*, Phoenix, Arizona, USA. IEEE, 197-200 Vol. 1.
- O'neil, T. J. and Manula, C. B. 1967. Computer simulation of materials handling in open pit mining. *Trans. AIME*, 238, pp 137-146
- Page J.R., E. H. 1994. *Simulation modelling methodology: principles and etiology of decision support*. Virginia Polytechnic Institute and State University.
- Panagiotou, G. N. 1999. Discrete mine system simulation in Europe. *International Journal of Mining, Reclamation and Environment*, 13, pp 43-46.
- Patten, B. J. 2009, Development of underground coal mine delay data analysis system, BE (Mining) Thesis, University of Wollongong, Australia.
- Pidd, M. and Carvalho, A. 2006. Simulation software: not the same yesterday, today or forever. *Journal of Simulation*, 1, pp 7-20.
- Porteous R. 2008. Fundamental Analysis of Development Cycles, Glencore, NSW.
- Porter, I., Baafi, E. Y. and Boyd, M. J. 2010. Underground roadway development delays. *In: Proceedings of the Third International Symposium Mineral Resources and Mine Development*, Aachen, Germany. pp 321-331.

- Porter, I., Cai, D. and Baafi, E. 2011. Using Flexsim simulation models to assess coalmine roadway development performance. *In: Proceedings of 35th APCOM Symposium*. Wollongong, Australia. 689-697.
- Prelaz, L., Bucklen, E., Suboleski, S. and Lucas, R. 1968. Computer applications in underground mining systems, Washington, D.C., Dept. of Engineering, Virginia Polytechnic Institute and State University.
- Rist, K., 1961. The Solution of a Transportation Problem by use of a Monte Carlo Technique, APCOM 1, University of Arizona, Nov.
- Robinson, S. 2004. Discrete-event simulation: from the pioneers to the present, what next? *Journal of the Operational Research Society*, 56, 619-629.
- Room and Pillar Mining [Online]. Available: <http://www.umwa.org/?q=content/room-and-pillar-mining> [Accessed 23.8. 2012].
- Sanford, R. L., 1965. Stochastic simulation of a belt conveyor system. Belt conveyor system, *In: Proceedings of APCOM 1965*, Tucson, AZ, USA. March, pp D1-D18.
- Schafrik, S. J. 2001. A new style of simulation model for mining systems. Master, Virginia Polytechnic Institute and State University, Blacksburg, VA, United States.
- Schafrik, S., Karmis, M., Agioutantis, Z. and Henderson, T. 2004. Computer simulation technology and demonstration. *In: Proceedings of the Thirteenth International Symposium on Mine Planning and Equipment Selection*, Wroclaw, Poland, 1-3 September. 605-612.

- Schwetman, H. D. 1990. Introduction to process-oriented simulation and CSIM. *In: Proceedings of the Winter Simulation Conference, New Orleans, Louisiana, USA.* 154-157.
- Shannon, R. E. 1998. Introduction to the art and science of simulation. *In: Proceedings of the Winter Simulation Conference. Washington, DC, USA,* 7-14.
- Sketchup website* [Online]. Available: <http://www.sketchup.com/products/sketchup-pro> [Accessed 23.6.2013].
- Simsir, F. and Ozfirat, M. K. 2008. Determination of the most effective longwall equipment combination in longwall top coal caving (LTCC) method by simulation modelling. *International Journal of Rock Mechanics and Mining Sciences*, 45, 1015-1023.
- Sturgul, J. 1999. Discrete mine system simulation in the United States. *International Journal of Mining, Reclamation and Environment*, 13, 37-41.
- Sturgul, J. R. 2000. *Mine design: examples using simulation*, Society for Mining Metallurgy, Littleton, CO, USA.
- Sturgul, J. R. 2001 Modelling and Simulation in Mining - Its Time Has Finally Arrived. *Simulation*, 76.
- Sturgul, J. R. and Smith, M. L. 1993. Using GPSS/H to simulate complex underground mining operations. *In: Proceedings of International Symposium Mine Mechanization and Automation, Lulea, Sweden.*
- Suboleski, S. C. and Lucas, J. R. 1969. Simulation of room and pillar face mining system. *In: Proceedings of the APCOM 1969, Salt Lake City, USA,* pp 373-384

- Topuz, E., Nasuf, E. and Ramachandran, D. 1989. ConSim: an interactive micro-computer program for continuous mining systems. *User Manual. Virginia Polytechnic Institute and State University. Blacksburg, VA.*
- Trueman, R. 1983. Master Thesis, *A study into the choice of mining methods for thin coal seams within the republic of South Africa.* University of the Witwatersrand, Johannesburg, South Africa.
- Underground Coal* [Online]. Available:
http://www.undergroundcoal.com.au/fundamentals/07_punch.aspx [Accessed 23.8.2012].
- Vagenas, N. 1999. Applications of discrete-event simulation in Canadian mining operations in the nineties. *International Journal of Mining, Reclamation and Environment*, 13, 77-78.
- OpenGL website* [Online]. Available at: <https://www.opengl.org/> [Accessed 16.11.2014].
- Zeigler, B. P., Praehofer, H. and Kim, T. G. 1976. *Theory of modelling and simulation*, John Wiley New York, USA.
- Zhao, R. and Suboleski, S. 1993. Graphical simulation of continuous miner production systems. *In: Proceedings of the Symposium on Emerging Computer Techniques for the Minerals Industry*; SME. Littleton, CO, USA. pp 343-354

APPENDIX A: FLEXSIM TECHNIQUES AND TERMINOLOGIES

Contents

APPENDIX A: FLEXSIM TECHNIQUES AND TERMINOLOGIES	260
1.1 THE TREE STRUCTURE OF FLEXSIM.....	261
1.2 TREE NODES.....	262
1.3 GENERAL RULES.....	263
1.4 MATHS OPERATIONS	264
1.5 VARIABLES COMPARISONS	266
1.6 LOGIC OPERATIONS	266
1.7 VARIABLE TYPES.....	267
1.7.1 <i>Single variables</i>	267
1.7.2 <i>Array variables</i>	267
1.7.3 <i>Declaring and setting variables</i>	268
1.8 EXECUTING COMMANDS.....	268
1.9 FLOW CONSTRUCTS	269
1.9.1 <i>If statement</i>	269
1.9.2 <i>While loop</i>	269
1.9.3 <i>For loop</i>	270
1.9.4 <i>Switch statement</i>	270
1.9.5 <i>Redirection</i>	271
1.10 TREE OBJECT REFERENCING	272
1.10.1 <i>Referencing commands</i>	272
1.10.2 <i>Data access</i>	275
1.10.3 <i>Node Ranks</i>	275
1.11 CODE COMMENTS AND EDITABLE DROPDOWN MENU	276

1.1 The Tree structure of Flexsim

The main sub tree nodes of the *project tree* (Figure 1) are as follows:

- exec: this tree contains simulation executive data that includes the simulation time, the event list, variables, functions, command list, Experimenter, OptQuest, and other information for running a model;
- library: the information for all *library* objects;
- model: the data for objects in the model built by the user, plus a Tools branch which contains *Flowitem Bin*;
- media: stores images, 3D models and sounds.

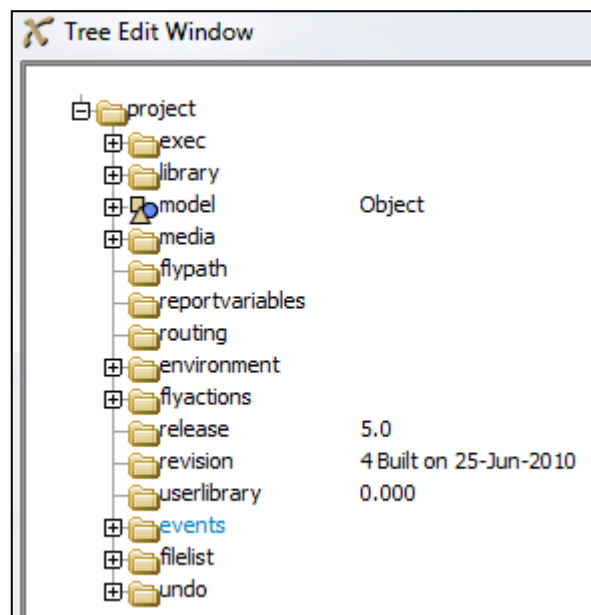


Figure 1 Project tree

The main sub tree nodes of the *view layout tree* (Figure 2) are as follows:

- active: every window opened by the user appears here;
- standard views: all the standard interfaces of Flexsim, including items such as the tree structure view, GUI builder, library view, charts view, and views of the tools;

- views library: the prebuilt view elements to build custom GUIs using Flexsim GUI builder;
- pages: the page views of object property setup interfaces;
- pick lists: all prebuilt pick lists used in Flexsim.

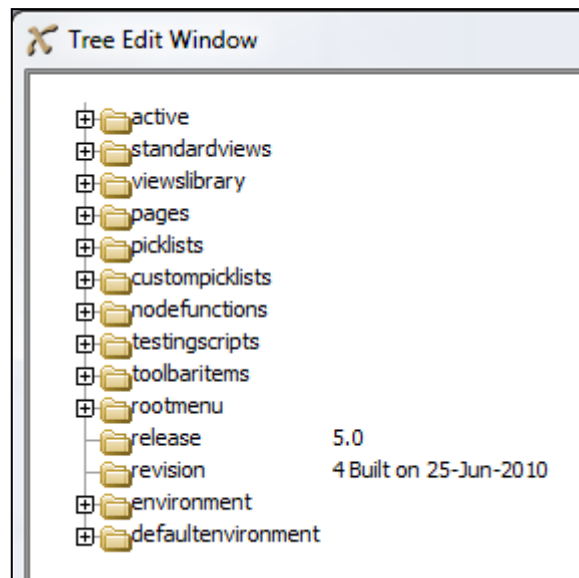


Figure 2 View layout tree





1.2 Tree nodes


Flexsim *trees* are made up of individual nodes that link together and hold information.

A node is a building block with a text name. There are five different types of nodes in Flexsim: standard, object, attribute/variable, function (C++) and function (*Flexscript*).

A node can be toggled to any one of the types through the *Build* menu or shortcut keys.


The symbols and usage for the different types of nodes are as follows:

- standard: , a standard node that holds data;
- object: , an object node;
- attribute/variable: , attribute /variable node that holds data;
- function (C++): , C++ node that holds C++ code;

- function (*Flexscript*):  , *Flexscript* node that holds *Flexscript* code.

Nodes can be a keyword used to define an attribute of an object or an item of data.

There are four data types: number, string (text), object or *pointer*.

Every kind of node can contain a sub list of nodes that are called the *content branch*. If a node contains sub nodes it can be expanded by clicking the  button. Nodes can be added and deleted from the tree.


An object node contains a second sub list of nodes called the *object attribute tree*. This separate tree branch contains data that describes the properties of the object with many special attribute nodes. When you click on an object node you see a greater than symbol  to the left of the node; clicking on this button opens the *object attribute tree* branch.

Figure 3 shows an expanded *object attribute tree* for a *processor* object in the model.

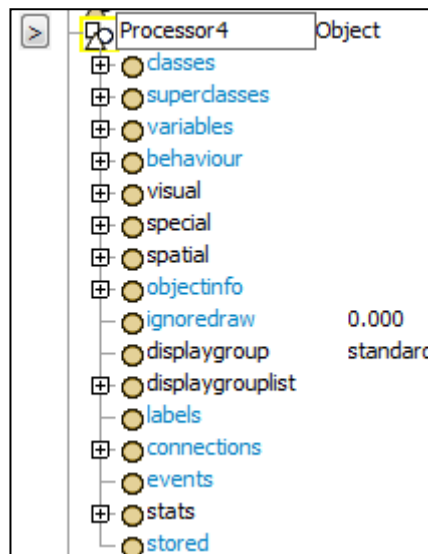


Figure 3 Attribute tree of a processor object

1.3 General rules

The following are some general rules for creating logic using *Flexscript*:

- language is case sensitive, i.e. Upper case A is not the same as lower case a;

- no specific format is required, the free use of spaces, tabs, and line returns is encouraged;
- numbers are double precision floating point values unless otherwise specified;
- text strings are usually entered between quotes. Example: "mytext";
- parentheses follow a function call and commas separate the arguments of the function. Example: `moveobject(object1,object2);`
- a function or command always ends with a semi-colon;
- parentheses can also be used freely to make associations in maths and logic statements;
- curly brackets are used to define a block of statements;
- to comment out the rest of a line use `//`;
- do not use spaces or special characters in name definitions. underscore (`_`) is ok;
- named variables and explicit values can be interchanged in a statement.

1.4 Maths operations

Table 1 lists common mathematical operations that can be performed on values.

Table 1 Maths operations

Operation	Floating Point Example (=solution)	Integer Example (=solution)
+	1.6+4.2 (=5.8)	2+3 (=5)
-	5.8-4.2 (=1.6)	5-2 (=3)
*	1.2 * 2.4 (=2.88)	3*4 (=12)
/	6.0/4.0 (=1.5)	20/7 (=2)
% (integer mod)		34%7(=6)
sqrt()	<code>sqrt(5.3)</code> (=2.3)	
<code>pow()</code>	<code>pow(3.0,2.2)</code> (=11.2)	<code>pow(3,2)</code> (=9)
round()	<code>round(5.6)</code> (=6)	
frac()	<code>frac(5.236)</code> (=0.236)	
fabs()	<code>fabs(-2.3)</code> (=2.3)	
fmod()	<code>fmod(5.3,2)</code> (=1.3)	

Table 2 shows typical assignment statement in Flexscript.

Table 2 Operations of setting and changing variables

Operation	Example
=	<code>x = x + 2;</code>
+=	<code>x += 2;</code> (same as <code>x = x + 2</code>)
-=	<code>x -= 2;</code> (same as <code>x = x - 2</code>)
*=	<code>x *= 2;</code> (same as <code>x = x * 2</code>)

<code>/=</code>	<code>x /= 2; (same as x = x / 2)</code>
<code>++</code>	<code>x ++; (same as x = x + 1)</code>
<code>--</code>	<code>x --; (same as x = x - 1)</code>

1.5 Variables comparisons

Examples of logical operations in *Flexscript* are shown in Table 3.

Table 3 Operations of variables comparison

Operation	Example (solution)
<code>></code> (greater than)	<code>1.7>1.7 (false)</code>
<code><</code> (less than)	<code>-1.7 < 1.5 (true)</code>
<code>>=</code> (greater than or equal to)	<code>45 >= 45 (true)</code>
<code><=</code> (less than or equal to)	<code>45 <= 32 (false)</code>
<code>==</code> (equal to)	<code>45 == 45 (true)</code>
<code>!=</code> (not equal to)	<code>45 != 35 (true)</code>
<code>comparetext()</code>	<code>comparetext(getname(current),"Processor5")</code>

1.6 Logic operations

Table 4 shows different operations for relating several comparisons.

Table 4 Logic operations

Operation	Example
<code>&&</code> (logical AND)	<code>x>5 && y<10</code>

(logical OR)	$x==32 \parallel y>45$
! (logical NOT)	$!(x==32 \parallel y>45)$
<code>min()</code>	<code>min(x, y)</code>
<code>max()</code>	<code>max(x, y)</code>

1.7 Variable types

Flexsim only uses four variable types, each of which can be used in an array structure.

The following explains each of these types.

1.7.1 Single variables

Table 5 shows the variable types used in Flexsim.

Table 5 Variable types

Type	Description
int	integer type
double	double precision floating point type
string	text string
treenode	reference to a Flexsim node or object

1.7.2 Array variables

Table 6 shows the array variable types used in Flexsim.

Table 6 Array variables

Type	Description
intarray	an array of integer types

doublearray	an array of double types
stringarray	an array of string types
treenodearray	an array of treeNode types

1.7.3 Declaring and setting variables

The following are some examples of how to declare and set variables.

```

int index = 1;

double weight = 175.8;

string category = "groceries";

treenode nextobj = next(current);

```

1.8 Executing commands

The procedure for executing a command in Flexsim is as follows:

- start with the command name, followed by an open parenthesis;
- next enter each parameter of the command, separating multiple parameters by commas; each parameter can be a variable, an expression of variables, or a command itself;
- finally, finish the command with a close parenthesis and a semi-colon.

Examples of execution commands are presented in Table 7.

Table 7 Example of executing commands

Syntax	Examples
<code>commandname(parameter1,parameter2,parameter3...);</code>	<pre> coloryellow(current); setrank(item, 3 + 7); setitemtype(item, getlabelnum(current, "curitemtype")); </pre>

1.9 Flow constructs

Flexscript uses similar conditional statement structures as in other languages. There are four conditional statements, **If** statement, **While loop**, **For loop** and **Switch** statements, and they can be combined to become a more complicated statement.

1.9.1 If statement

The **if** statement allows one piece of code to be executed if an expression is true, and another piece of code if it is false. The **else if** and **else** of the construct are optional, and one can have as many **else if**'s as required but only one **else** can be used. The construct is shown with an example in Table 8.

Table 8 Construct and example of **If** statement

Construct	Examples
<pre>if (test expression) { code block } else if { code block } ... else { code block }</pre>	<pre>if (content(item) == 2) { colorred(item); } else if { colorblack(item); } ... else { code block }</pre>

1.9.2 While loop

The **while** loop statement continues looping through its code block until the test expression is recognised as false. Table 9 shows the **While** loop construct with an example.

Table 9 Construct and example of **While** Loop statement

Construct	Examples
<pre>while (test expression) { code block }</pre>	<pre>while (content(current) == 2) { destroyobject(last(current)); }</pre>

1.9.3 For loop

The **for** loop is similar to the **while** loop, but it is usually used when it is known exactly how many times to loop through the code block. The start expression is executed only once, to initiate the loop. The test expression is executed at the beginning of each loop and the loop stops as soon as this expression is false, just like the **while** loop. The count expression is executed at the end of each loop, and typically increments a loop variable, signifying the end of one iteration. Table 10 shows the **For** loop construct with an example.

Table 10 Construct and example of **For** Loop statement

Construct	Examples
<pre>for(start expression; test expression; count expression) { code block }</pre>	<pre>for (int index = 1; index <= content(current); index++) { colorblue(rank(current, index)); }</pre>

1.9.4 Switch statement

The **switch** statement allows one piece of code to be used to execute from several possibilities, depending on the value of a variable being switched on. The **switch** variable must be an integer type. The example shown in Table 11 sets the colour of items of type 1 to yellow, type 2 to red, and all other types to green.

Table 11 Construct and example of **Switch** statement

Construct	Examples
<pre> switch (switchvariable) { case casenum: { code block; break; } default: { code block; break; } } </pre>	<pre> int type = getitemtype(item); switch (type) { case 1: { coloryellow(item); break; } case 2: { colorred(item); break; } default: { colorgreen(item); break; } } </pre>

1.9.5 Redirection

Each of the above statement flows can be redirected during execution with either a **continue** statement, a **break** statement, or a **return** statement. Table 12 explains how each of these statements works.

Table 12 Redirection statements

Construct	Examples
continue;	Only valid in For and While loops. Halts the current iteration of the loop and goes on to the next iteration in the loop. In a For loop the counter is incremented before continuing.
break;	Only valid in For , While and Switch statements. Breaks out of the current For , While or Switch block and continues with the line immediately following this block. Nested statements only break out of the current statement and continue on in the containing statement.

return;	Returns out of the current method entirely and continues with the line following the code that is called this method. A return value may be required if the method returns a value.
----------------	--

1.10 Tree object referencing

When all the model information is stored in a Flexsim tree access must be gained in order to make changes such as node attributes using *Flexscript*. There are many ways to gain access to tree nodes, node names, and values (attributes), and ways to set node information such as name and data value.

Table 13 summarises *Flexscript* referencing commands

1.10.1 Referencing commands

Table 13 Examples of referencing commands

Command(parameter list)	Explanation	Example
c	This is used to reference the active node during a function's execution. If the function was called using nodefunction(), c returns a reference to the node on which the function is written. If the function is an event function, c returns a reference to the object that contains the event.	<pre>current= ownerobject(c);</pre> <p>So, current is a reference to the object that owns the node.</p> <pre>item=parnode(1);</pre> <p>So item is a reference to the involved <i>flowitem</i> for a trigger or function.</p>
i	This is used only in event functions. It is used to reference the object that was passed to the function as the involved object.	<pre>getname(i);</pre>
<pre>first(node) next(node) last(node)</pre>	This returns a reference to the <i>first</i> (last, next) ranked object inside of the object passed	<pre>first(current); next(item); last(current);</pre>

<code>node(relativepath[,startnode])</code>	This returns a pointer to the node in the tree that is found by following the path specified as <code>relativepath</code> , starting at the node specified as <code>startnode</code> .	<code>node("/Processor", model());</code>
<code>main()</code> , <code>model()</code> , <code>views()</code> , and <code>library()</code>	These are commonly used as the start node of node function, refer to the project tree, model tree, view tree, and library tree respectively.	<code>main();</code> <code>model();</code> <code>views();</code> <code>library();</code>
<code>inobject(object,portnum)</code> <code>outobject(object,portnum)</code> <code>centerobject(object,portnum)</code>	This returns a reference to the object connected to the input(output, centre)port number of the object passed	<code>inobject(current,1);</code> <code>outobject(current,1);</code> <code>centerobject(current,1);</code>

The attribute/variable nodes of the Flexsim tree in blue (Figure 4) can be used as a command to access the node. For example, `variables(longwall_conveyor)` returns the variables node of a `longwall_conveyor` which must be a reference to the longwall conveyor treenode; `spatialx(face)` returns the node that stores the width of the longwall face because `face` is a reference to the longwall face object in the model.

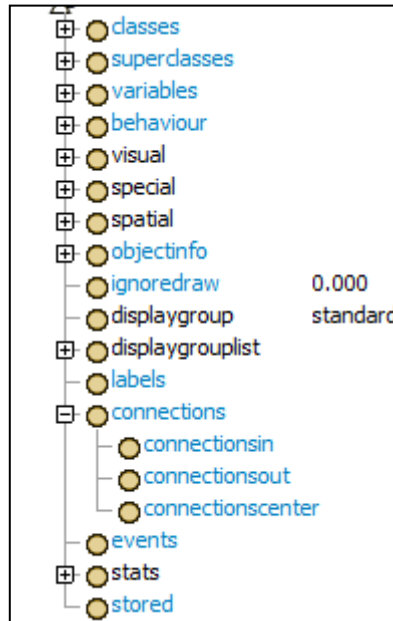


Figure 4 Flexsim attribute tree with nodes in blue

Unlike unix or windows, Flexsim uses a path syntax for file paths. There are several path symbols that can be used to traverse the tree, as listed below:

- / go into the current node's sub-tree;
- > go into an object's attribute tree;
- .. go to the current node's parent tree, or up one level;
- @ go to the owner view of the current node;
- + read the current node's text as a path to an object, and go to that object.

For the most part, only the / and > symbols are used, and occasionally the .. symbol.

The @ and + symbols are only used to build a custom GUI. For example:

the value of the *objectfocus* node of the shearer's customised GUI is "MAIN:/project/model/working_face", while the link of an edit box on the GUI is "@>objectfocus+/Shearer>labels/2". The part "@>objectfocus+" equals the value in the *objectfocus* node. Then the link is the same as "MAIN:/project/model/

`working_face/Shearer>labels/2`” which refers to the second label of the Shearer object which contains the shearer task table.

1.10.2 Data access

In Flexsim the values for objects are mostly obtained using commands starting with “get”, while the node values are mostly set by using commands starting with “set”. These are the two groups of commands that are most frequently used. Commands for other functions can easily be found by the name of the function in the commands reference manual, provided the command required is known in the Flexsim terminologies. Table 14 shows the examples of data access commands.

Table 14 Examples of data access commands

command(parameter list)	Explanation	Example
<code>get????(object, parameter list)</code> <code>get??num(object, parameter list)</code> <code>get??str(object, parameter list)</code> <code>get??name(object, parameter list)</code>	This returns the number/string/name value of the variable with the given parameters	<code>Getname; getoutput;</code> <code>Gettablenum; getnodenum,</code> <code>getnodestr; getvarstr;</code> <code>getnodename;</code>
<code>set????(object, parameter list)</code> <code>set??num(object, parameter list)</code> <code>set??str(object, parameter list)</code> <code>set??name(object, parameter list)</code>	This sets the number/string/name value of the variable with the given parameters	<code>setcolor; setrank;</code> <code>setlabelnum;</code> <code>settablestr;</code> <code>setnodename;</code>

1.10.3 Node Ranks

In a Flexsim tree the nodes are arranged in a specific order (Figure 5). To show their ranks, right click at any part of the tree view and then select **View | Show Node Ranks**.

In *Flexscript*, a node can be referred to by the node name or the node rank. Using the node name can make the code more readable but may slow the model execution down because Flexsim must compare each of the nodes in the list from the top with a specific name until it is found. Flexsim directly executes the one required if using the rank index, but this can lead to confusion by the numbers over time. The recommended way to refer to a node is to use rank values and put comments at the end beginning with the symbol `//`.

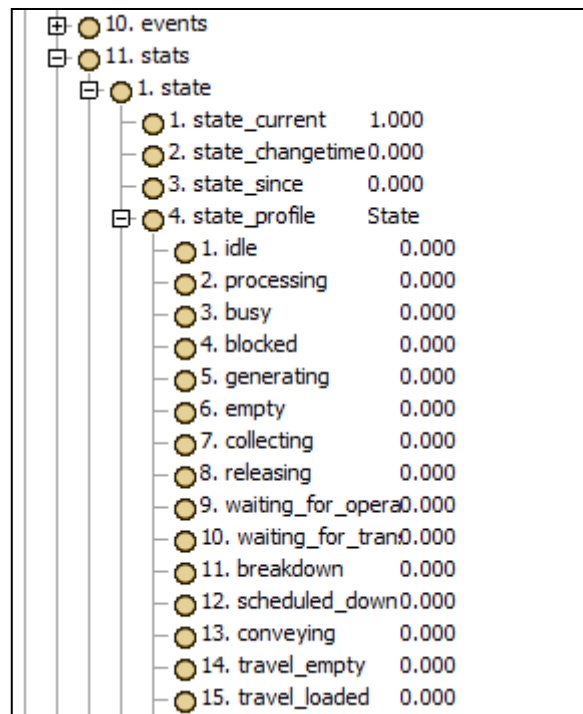


Figure 5 Node ranks


1.11 Code comments and editable dropdown menu

There are two ways to document comments in *Flexscript*, the first is the one-line comment which is done with two forward slashes: `//`. The example below shows a one-line comment

```
// this is my one-line comment, it ends at the end of this line
```

The other comment is a multi-line comment where the start of the comment is signalled with the combined symbol `/*` and the end with the symbol `*/` as shown below:

```
/*
this is my multi-line comment
it can span as many lines
as I want it to
*/
```

In the Flexsim template code some text can be edited in a drop-down when  is clicked on. The text is split into black text which is fixed and blue text which can be edited (left part of Figure 6).

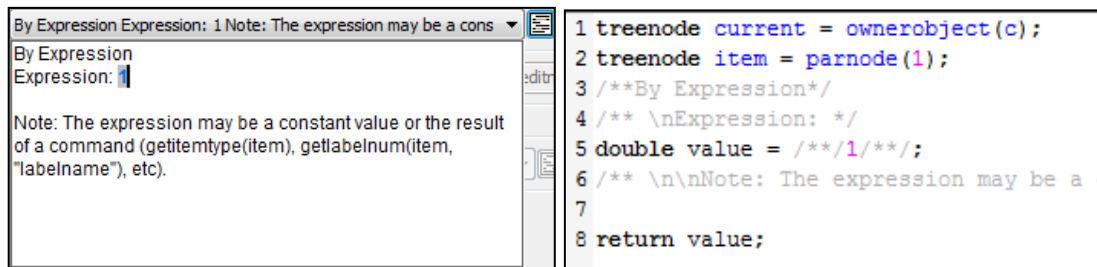




Figure 6 Drop-down menu with editable text and the template code

In the template code fields (the right part of Figure 6 can be opened by clicking on the  symbol), variables called `current` and `item` can be seen: “current” refers to the object whose code is being edited, and “item” refers to a *flowitem* associated with a specific execution of the field. These two variables are common for most template codes.

In the template code the grey parts are comments that only the *Flexscript* executer sees: `double value = 1`. In this format the dropdown menu is as shown on the left of Figure 6 with editable blue text. The advantage here is that any user can quickly change the *1* value to something else by pressing the button  and editing the blue template text.

The functions are as follows:

- `bernoulli(prob, succeed-value, fail-value, stream);`
- `beta(min, max, shape1, shape2, stream);`

- `binomial` (trials, prob, stream);
- `cempirical` (tablename, stream);
- `dempirical` (tablename, stream);
- `duniform` (min, max, stream);
- `empirical` (tablename, stream);
- `erlang` (location, scale, shape, stream);
- `exponential` (location, scale, stream);
- `gamma` (location, scale, shape, stream);
- `geometric` (prob, stream);
- `inversegaussian` (location, scale, shape, stream);
- `invertedweibull` (location, scale, shape, stream);
- `johnsonbounded` (min, max, shape1, shape2, stream);
- `johnsonunbounded` (location, scale, shape1, shape2, stream);
- `loglaplace` (location, scale, shape, stream);
- `loglogistic` (location, scale, shape, stream);
- `lognormal` (location, scale, shape, stream);
- `lognormal2` (location, scale, shape, stream);
- `negbinomial` (successes, prob, stream);
- `normal` (mean, stddev, stream);
- `pearson5` (location, scale, shape, stream);
- `pearson6` (location, scale, shape1, shape2, stream);
- `poisson` (mean, stream);
- `triangular` (min, max, mode, stream);
- `uniform` (min, max, stream);

➤ `weibull` (location, scale, shape, stream).

The “stream” parameter in the commands is a reference to one of Flexsim's random number streams; for more information, please refer to Flexsim commands help manual (2013).

Three of these functions are worth discussing in some detail: `empirical`, `cempirical` and `dempirical`; `empirical` and `cempirical` are both continuous empirical distributions but return different ranges of continuous numbers, while `dempirical` is a discrete empirical distribution that returns explicit values listed in the table. These three functions can be used when the data sample is small or when no distribution function fits with the sample data.

In Table 15 the section on the left is an example of reference tables created by the user with 4 rows and 2 columns. In column one the percentages are 10, 20, 30 and 40, which add up to 100 per cent. In column two the values are 0.1, 0.2, 0.3 and 0.4. The table may have as many rows as needed to define as many values as desired.

Table 15 Summary of the three distribution functions

Reference Table (table name: mytable)		Possible Return Values ("X")		
Percentage	Values	Dempirical (mytable)	Empirical (mytable)	Cempirical (mytable)
10	0.1	$X = 0.1$	$0.1 < X \leq 0.2$	$0.0 < X \leq 0.1$
20	0.2	$X = 0.2$	$0.2 < X \leq 0.3$	$0.1 < X \leq 0.2$
30	0.3	$X = 0.3$	$0.3 < X \leq 0.4$	$0.2 < X \leq 0.3$
40	0.4	$X = 0.4$	No value	$0.3 < X \leq 0.4$

The right hand section of Table 15 lists the possible return values of the three commands. The discrete command only generates random variates that exactly match the values entered into column two of the table at the specified possibility. The continuous commands generates random variates that are real numbers distributed uniformly between two adjacent numbers in column two of the table. The difference between the two continuous commands is how the bounds of the uniform ranges are defined.