QUALITY CONSTRAINED SCHEDULING
OF MINING OPERATIONS

A Thesis Presented to
The Faculty of the Russ College of Engineering and Technology
Ohio University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science

By
Yang Bai

August, 1994
ACKNOWLEDGMENT

I would like to express my sincere appreciation to Professor Donald E. Scheck, Ph.D., who has advised me throughout the preparation of this research and has provided his lab facilities, ICAMPS™ LIB routines and WATCOM™ C compiler for my research. Appreciation is extended to Professor Robert L. Williams, Ph.D. and Professor Robert P. Judd, Ph.D., members of my thesis committee, for their interest and support.

I also appreciate the three years of invaluable education and support for my research extended to me by the Department of Industrial and Systems Engineering.

A special gratitude must be given to my parents in The People's Republic of China, Zhenhua Bai and Kexiang Sun, whose support and encouragement have been constant and inspiring, and also to my wife Min Wang and my beloved daughter Yiqun Bai, for their understanding and patience throughout the challenging years.

Finally, I would like to give my special thanks to Kevin Cunningham, a computer specialist in LINDO Systems Inc., who has given me the valuable support for tailoring LINDO library functions to meet my special research requirement.

Yang Bai
Athens, Ohio, USA
ABSTRACT

This study endeavored to develop an algorithm named Quality Constrained Scheduling of Mining Operations (QCSMO) for underground coal mining. This algorithm uses dynamic data structures and mixed linear-integer programming (MLIP) technique to develop computer software that generates a mining sequence such that the mining units and areas are scheduled within a given time horizon to meet both coal production and quality requirements. QCSMO guarantees that the scheduled sequence is optimal within each period and at least "near optimal" throughout the whole process, under the spatial and operational precedent constraints.

QCSMO dynamically builds MLIP models and uses LINDO™, a linear optimization software, to optimally assign mining units to areas within each period. The final output of QCSMO can help the tasks of generating a mining plan, estimating the quality of production or evaluating a mining layout. The dynamic nature of QCSMO allows users to adjust production requirements at any period within the time horizon and gives them much flexibility to fit the needs in the real world.

The implementation of this algorithm required inter-language calls between C and FORTRAN routines. The method used by this research may be applied to other engineering fields where operational process control and optimization are needed and when productivity and quality are concerned simultaneously under various precedent constraints of activities.
TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acknowledgment</td>
<td>iii</td>
</tr>
<tr>
<td></td>
<td>Abstract</td>
<td>iv</td>
</tr>
<tr>
<td></td>
<td>Table of Contents</td>
<td>v</td>
</tr>
<tr>
<td></td>
<td>List of Illustrations</td>
<td>vi</td>
</tr>
<tr>
<td>1.</td>
<td>INTRODUCTION</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.1 Statement of the Problem</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1.2 Literature Review</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1.3 Problem Description</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>1.4 Constraints and Requirements on Mine Scheduling</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>1.5 Objectives of QCSMO</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>1.6 The Justification of Optimality</td>
<td>9</td>
</tr>
<tr>
<td>2.</td>
<td>DISCUSSION OF QCSMO ALGORITHM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.1 Flow Diagram of QCSMO</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>2.2 QCSMO User Interface</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>2.3 QCSMO ICAMPS Interface</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>2.4 QCSMO LINDO Interface</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>2.5 Processing Steps in QCSMO</td>
<td>16</td>
</tr>
<tr>
<td>3.</td>
<td>BUILDING THE MLIP CONCEPTUAL MODEL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.1 Symbols and Terminology</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>3.2 Variables Definition</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>3.3 MLIP Model</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>3.3.1 MLIP Objective Function</td>
<td>21</td>
</tr>
</tbody>
</table>
3.3.2 MLIP Primary Constraints
3.3.3 MLIP Dynamic Constraints
3.4 Summary of MLIP Dynamic Constraints

4. A CASE STUDY BY APPLYING QCSMO

4.1 Case Description
4.2 Initializing QCSMO Scheduling
4.3 Schedule Period by Period
4.4 The Result of QCSMO Scheduling

5. CONCLUSIONS

5.1 Further Discussion on QCSMO
5.2 Comparing QCSMO with ICAMPS
5.3 Expansion of QCSMO

BIBLIOGRAPHY

APPENDIX A Inter-Language Calls Between C and FORTRAN
APPENDIX B QCSMO Input and output Files
APPENDIX C Program Listing
APPENDIX D QCSMO Supporting Programs
LIST OF FIGURES

Figure 1.1 Typical mine layout. 6
Figure 1.2 Grid superimposed on the mining layout. 7
Figure 2.1 Flow Diagram of QCSMO Algorithm 12
Figure 3.1 Precedent relationship between a LW area and its surrounding DV areas. 21
Figure 3.2 Mining Sequence Requirements. 25
Figure 3.3 Remaining tonnage relationship of LW1 and the DV areas of LW2. 26
Figure 3.4 Possible CM Mining Options. 30
Figure 4.1 Mine Layout of XYZ Coal Co. 33
Figure 5.1 Gantt Chart showing ICAMPS and QCSMO mining sequences of DV areas mined by CM1 and CM2. 52
Figure 5.2 Example where two LW areas are mined during the transition period. 55
Table 5.1 Comparison between ICAMPS and QCSMO for Whole Mine 49
Table 5.2 Comparison between ICAMPS and QCSMO for Steady State 50
1.1 STATEMENT OF THE PROBLEM

In coal-mining industry, mining planning and scheduling are key considerations of management. Planning and scheduling, both long-term and short-range, are playing an increasingly important role in decision making. In today's rapidly-changing world economy and environment considerations, coal is no longer considered solely as a pure raw material and energy resource, but also a source of potential pollution and environmental damage. Coal quality becomes a dominant factor in coal exploration and production.

Coal is used not only as a fuel for power generating but also as a raw material for various metallurgical and chemical processes. The different usages of coal have various quality requirements. Today, the issue of environmental protection and preservation are placing more and more restrictions on the use of coal and its byproducts. A coal producer may not be able to stay in business unless considerations are taken beforehand about the quantity and quality of production. Some coal mines have shut down because they could not produce coal to meet rigid quality requirements at an economical operating cost.

Many parameters determine the quality of coal. The most important ones are the amount of sulfur per B.T.U., the percentage of ash and moisture content per pound and the calorific value expressed in B.T.U. per pound.

Since coal is a naturally occurring-material, the constituents of coal are determined largely by the geological environment during deposition. The most common way to control the quality of coal is by blending, that is, mixing different grades of coal to create a mixture which meets the quality specifications.

The linear programming (LP) model for coal blending is a well-known technique and has been used in the industry for more than thirty years (Mark Gershon, 1990). It is used
not only to control the coal quality but also to make full use of this valuable and non-renewable natural resource.

An LP blending model can be easily solved by a linear optimizer such as LINDO™ (Linear, INteractive, and Discrete Optimizer). The final solution will determine the optimal amount of each grade of coal available to meet the given quantity and quality requirements at a minimum cost [LINDO Manual, 1991].

Because of geological variation within coal deposits, the composition of coal may vary from place to place and from seam to seam. The quality distribution of a coal deposit is always affected by this anisotropical nature.

Since the quality of coal varies within a deposit and coal with the required quality is not always accessible, some mines must create stockpiles of coal of different qualities and blend them into an acceptable product. To ensure an adequate supply, a large amount of coal has to be stored for blending purposes. Stockpiling incurs extra costs for inventory investment and handling as well as an increase in the likelihood of spontaneous combustion of the stockpile. If there is a shortage of the needed grades, the blending process would be halted until an external resource is obtained. Blending coal on the surface costs coal producers a considerable amount of money due to the extra effort required to manage and handle the coal inventory and ensure its safety.

If a coal producer could do this blending at the mining faces, that is, while mining, the quality of coal coming out of the mine could meet specifications without the need for further processing. By correctly selecting the cutting faces and maintaining the necessary production from each mining unit, the coal produced by mixing current production could meet or at least be very close to the quality requirements so that the need for surface blending would be reduced to a minimum. By doing so, the land for storing coal would be saved and the life of a coal mine would be extended due to better utilization of the available resources.
1.2 LITERATURE REVIEW

It is not a new idea to plan and schedule mining operations with the consideration of productivity and quality. However, because of its complexity, a general solution has never been developed. The research efforts on this problem have concentrated on two approaches. One focused on the theoretical optimality of the scheduling while the other on the practical implementations.

The theoretical school of thought achieves optimal solutions by using Linear Programming (LP) and Integer Programming (IP) [Johnson, 1969, Gershon, 1982], but the model assumptions are too restrictive to be practical. Few if any of the theoretical models have been implemented, but they are very important in the sense of guiding the direction of the development of practical applications.

On the other hand, the practical school of thought developed heuristic approaches, most of which are based on computer simulation and trial and error. Many mining engineers and experts have endeavored to conceive methods and algorithms that can obtain so-called "near-optimal" solutions. A near-optimal solution, sometimes called a "heuristic solution", is acceptable where no optimal solutions can be found or when an optimal solution is too difficult to obtain. Heuristics can provide a very effective way to acquire near-optimal solutions that can, in most of cases, meet the practical needs of production planning and scheduling. Although the heuristic approaches provide solutions with unknown optimality; heuristics are appealing because they are easy to use.

In 1986, a student at Ohio University, Ilango Sankaralingam [Sankaralingam, 1987] developed a heuristic algorithm. It can generate a mining sequence that satisfies production tonnage requirements over time, while minimizing the time to access long wall areas and maximizing the total mine production by increasing the opportunities to open up more mining areas. A long wall unit is crucial since it is the most productive and expensive equipment in mining. Because of the complexity of underground mining
scheduling, this algorithm succeeded in generating a workable mining sequence under various operation constraints, but it did not consider the coal quality. In 1991, Eddy Herjanto [Herjanto, 1991] developed an underground mining planning and scheduling technique that considered both productivity and quality considerations. His algorithm generated a sequence of areas to be mined that meets specific quality and production requirements. His LP-based algorithm is most applicable to designing and planning a new mine before any real mining takes place. Under most conditions, his approach can generate a near-optimal solution. The quality of coal mined by following his sequence will require minimal blending to meet the specification and avoids producing a better than necessary product. High and low quality areas are mined proportionally to the quality requirements to make full use of the most valuable natural resource.

An extensive search of the literature, did not turn up any algorithm that can optimize the whole scheduling process, especially, when the quality of coal production is considered. Herjanto's thesis contains an extensive list of methods and algorithms for developing mining scheduling. Only a few are relevant to this research.

Two doctoral dissertations from the University of Arizona dealt with producing coal to meet quality specification. Baafi (1983) used mathematical programming to solve the coal blending problem and three models were developed for blending a combination of coal on the surface, produced underground and purchased. Cost was also considered. In his first model he tried to use 0-1 programming to assign mining units to working benches such that the quality of coal would yield high thermal content and low pollutants. His research differs from the model presented in this thesis. The model he used is a one time global assignment and underground mining precedence relationships are not considered. Lonergan (1983) generated a computerized mining planning system that can help the coal suppliers to predict the coal quality of future shipments by proper planning of underground coal mining. Neither dissertation addressed the problem of an operational
Bernardo (1988) addressed the application of linear programming in underground coal mining scheduling. He developed two models that dealt with the aggregate planning process and two-machine scheduling. He realized that the underground coal mining scheduling was very complex because of the rigid time and spatial constraints upon the objects to be scheduled, so he used heuristics to solve the problem. Another researcher, Paul Appiah (1989), used Pareto optimal stochastic serial dynamic programming to help miners in making decisions on various planning considerations. Gershon (1986) described an approach called "Blending Based Approach to Mine Planning and Scheduling." He discussed in general the requirements and theoretical fundamentals of the problem but did not provide a practical algorithm to implement his idea.

1.3 PROBLEM DESCRIPTION

Because coal is a naturally occurring raw material with spatial variation in quality and the coal market changes rapidly, the mining production and quality requirements are always changing. The need to adjust the existing mining plan and rearrange mining areas to meet the challenge of new situations is inevitable. The algorithm developed by Herjanto does not apply to the on-line or real time planning and scheduling. His algorithm provides information which can be the basis for a mine layout, but it does not consider an actual mine layout. The shortcoming of his method restricts its application to long term planning. A new algorithm, therefore, is needed to schedule mining operations dynamically according to the changing constraints on productivity and quality. This research called Quality Constrained Scheduling of Mining Operations (QCSMO) addresses the problem by building on the previous research. The QCSMO algorithm developed in this research must generate an implementable and near-optimal mining sequence that can be directly used as an operational production plan and be easily
modified according to changes of production requirements. This dynamic nature provides flexibility for both long-term planning and short-range day-to-day scheduling.

1.4 CONSTRAINTS AND REQUIREMENTS ON MINE SCHEDULING

The QCSMO algorithm is based on following assumptions and premises. Two kinds of production equipment are used in underground coal-mines. One is the continuous miner (CM) and the other is the long wall miner (LW). A coal mine is composed of blocks of coal and each block is divided into areas that reflect the mining method in use. The area to be mined by a long wall unit is called a long wall area and is also denoted by LW. Areas mined by CM units are called development areas (DV). Figure 1.1 shows a typical mine layout.

![Figure 1.1 Typical mine layout.](image-url)
A LW area is surrounded by DV areas and can be mined only after all its surrounding DV areas have been mined, because a LW area must use the development areas on its boundaries for ventilation, material handling and other facilities. DV areas are mined by a method called room and pillars that leaves part of the coal as supporting pillars to prevent the roof of the mining area from falling.

In order to develop a schedule to meet the quality requirements, the contents of the coal and geological parameters such as seam thickness, specific gravity and chemical contents must be estimated. All the required information is stored in a file called a grid which is continuously updated with the new information from mining and exploration activities. The grid generally consists of a point estimate in a rectangular pattern as shown in Figure 1.2.

Figure 1.2 Grid superimposed on the mining layout.
Each grid point is assumed to represent the quality of the surrounding area, such as the seam elevation, thickness and average ash, moisture, sulfur and B.T.U. content. The grid spacing should reflect the available information. If the sample data is very sparse, only a very coarse grid can be justified.

1.5 OBJECTIVES OF QCSMO

The QCSMO algorithm can handle any time period like days, weeks, months or other specified interval. For each period, the requirements, i.e., the total tonnage and the coal quality standards, are provided by mine management. These requirements need not to be uniform throughout the entire planning horizon.

The major objective of the QCSMO research was to develop an algorithm that generates a mining sequence, that is, to specify when and where a mining unit should work and how much production should come from the assigned area(s) subject to the quantity and quality constraints.

In general, the production of the LW units contributes a major portion, usually about eighty percent, of the total production. In addition, once a LW unit begins to mine an area, it must stay in that area until it is completed. Therefore, the coal quality produced by LW units tends to determine the final quality in a given period. Although the CM units usually mine the surrounding DV areas that provide access to the LW areas, a CM unit can produce in several areas within the same period. In a sense, the production of the CM units provides an adjustment factor to the long wall coal quantity and quality.

The final production schedule should fulfill the following objectives in hierarchical order of their importance.

1. In each scheduling period, the cumulative tonnage of coal produced by the units should meet the production requirement of that period.

2. During the entire scheduling time horizon, a LW unit should always have an area
to mine unless the LW unit needs to be moved to a new area or all the LW areas have been mined. Also, at the very beginning of the scheduling, no LW areas may be accessible.

3. In each of the scheduling periods, the blended coal produced in that period should have minimal deviation from the specific standard. Any deviation will be eliminated by blending coal from an external source.

4. During the scheduling process, subject to the restriction that no LW unit be delayed, a CM unit should be idle as much as possible to prevent premature investment in long wall area development.

5. For life-of-mine scheduling, the entire resource of coal should be exhausted.

1.6 THE JUSTIFICATION OF OPTIMALITY

Since this research deals with practical underground coal mining scheduling, the technological considerations of coal mining will affect the optimality of scheduling. These considerations include the following:

1. The non-violable precedence relation among areas. That means there is no possibility to mine areas selectively in underground mining. Any area to be mined must be accessible from an adjacent previously mined area.

2. The acceptability of near-optimal solutions. A schedule is said to be near-optimal if in each period the coal produced meets the quantity and quality requirements within the range of acceptable variation and, by the end of schedule, all resources are exhausted. Though this algorithm can not guarantee the global optimality, however, in practice the long term constraints on production are not known. A global optimum solution would only be as good as the long term assumption about production requirements.

3. The spatial distribution of the coal quality is considered to be independent within the coal seam.
4. Coal production contracts determine the specifications about coal quality. Such contracts usually do not cover the entire life of the mine.

5. The hierarchical satisfaction of quality specifications. The quality requirements are met based upon a user-defined penalty for each quality deviation. The objective function, which is a summation of weighted deviations away from requirements, is minimized within each period. Each individual deviation is not necessarily minimized though a global optimum may have been reached.
CHAPTER II
DISCUSSION OF QCSMO ALGORITHM

2.1 FLOW DIAGRAM OF QCSMO

QCSMO algorithm is composed of four parts, the main program, QCSMO user interface, ICAMPS (Interactive Computer Aided Mine Planning System) interface and LINDO interface. Figure 2.1 shows the fundamental structure and basic logic flow about this algorithm.

QCSMO itself is not a stand-alone system, instead it needs to be embedded into another host program. In this research, ICAMPS is the host of QCSMO and it provides all input and output services for QCSMO. The support from ICAMPS is vital to this research, but in later implementations of QCSMO, the user can choose any platform in which to embed QCSMO as long as the host provides all the I/O services and information needed by QCSMO.

As discussed below, one of the main contributions of QCSMO is its dynamic interfacing with LINDO. QCSMO imbeds LINDO and uses it for optimizing a mixed linear and integer programming (MLIP) model which is the heart of QCSMO. MLIP is discussed in detail in Chapter III. The user of QCSMO does not need to know how LINDO works and how to interface it.

2.2 QCSMO USER INTERFACE

The user interface of QCSMO receives all the user requirements and other information needed for scheduling. The objective file contains the scheduling objectives, such as production requirements, the quality requirements and the penalty coefficients for each deviation from the requirements. The grid file provides the geological information about the mine. The area definition file gives the area spatial relations such as the
Figure 2.1 Flow Diagram of QCSMO Algorithm
sequence in which the LW areas will be mined. During the scheduling process, QCSMO saves the identification numbers of all scheduled areas and other relevant information into the GC file, which contains the final result from QCSMO. The GC file records are in chronological order, and include the mined area's identifier, the tonnage mined and quality deviations by area. The user can use the GC file information to draw a Gantt Chart, a map of the mining plan or tailor it for other planning purposes.

2.3 QCSMO ICAMPS INTERFACE

Much data manipulation is performed by the functions from the ICAMPS library. Those functions perform the basic reading and writing tasks for grid file, area definition file and other related files which are needed for scheduling. Another important function of the ICAMPS library is to interpret the quality parameters from a grid file. All the geological information stored in the grid file is in binary format and needs to be translated by the ICAMPS routines before QCSMO can use it.

All data files in ICAMPS are indexed. An indexed file is required to manage a data file with a large amount of records and to manipulate retrieve, update, insert and delete information. ICAMPS uses an indexed file managing system call BTree/ISAM, which is a commercial software product from Softfocus of Canada. ICAMPS embedded this system into its own environment. Because Btree files are in binary format, they cannot be listed without converting to the normal ASCII format. A special program named LOOK.EXE was created to convert the two QCSMO working files, WORK.QP and WORK.GC, into ASCII format. The Btree file managing system is very fast and efficient for small scale database management. The interested reader may refer to the BTree/ISAM V3.1 manual for more information.
2.4 QCSMO LINDO INTERFACE

The main difficulties encountered in communicating with LINDO was caused by the language differences. LINDO is written in FORTRAN. QCSMO is written in C because C has more powerful features for handling dynamic data structures. Putting modules written in different languages together is restricted not only by the function-calling conventions among languages but also by the differences among working platforms. Protocol must be set before routines in one language can call routines in the other. In order to combine FORTRAN and C together, the author chose WATCOM C as the host programming language and LINDO SYSTEMS Inc. converted LINDO to WATCOM FORTRAN. Since both modules are on the WATCOM platform, they can call each other through built-in PRAGMA directives. The author, with the help of LINDO SYSTEMS Inc., spent a considerable amount of time and effort on this issue but finally succeeding in calling LINDO subroutines from QCSMO C functions. The details of this calling procedure appear in Appendix A. Those who are interested in calling FORTRAN from C and vice versa may find this information helpful in other applications.

Another difficulty in implementing the QCSMO algorithm is dynamically updating the MLIP model in accordance with the changing conditions of the mining units and working areas. The MLIP model needs to be modified whenever the status of a mining unit changes. In order to do so, a linked list data structure named ARELIST, is used to store the schedule data for the MLIP model. When a new area is introduced into the schedule, it is appended to the tail of the ARELIST and when an area is completed, it is dropped from the list. Therefore, the ARELIST always contains the areas that are currently being mined or waiting to be mined. The stopping criterion for the QCSMO is quite simple, when ARELIST is empty, scheduling ends.

The MLIP model is composed of primary and dynamic constraints. As shown in the Chapter III, the primary constraints control the parameter deviations and are required for
every period, while the dynamic constraints are chosen by checking the mining status of
each area in the ARELIST. The MLIP model builder checks which areas are currently
being mined, waiting or completed, then the corresponding dynamic constraints are
brought into the current MLIP model. After all the areas in the list are checked, the MLIP
model is saved in MPS format in a file named LNDIN which LINDO can read.

The MLIP model is solved by calling LINDO to perform linear optimization. The
QCSMO algorithm calls LINDO each period iteratively as it generates the mining
sequence. Since loading the LINDO software into the computer memory is very time
consuming, calling LINDO at the very beginning of the scheduling and allowing it to
reside in memory are mandatory for a practical application of this algorithm. The detailed
procedure of interfacing with LINDO is shown in the APPENDIX A.

The QCSMO uses two file formats to interface with LINDO. The input is in MPS
format, a commonly used format in industry for describing a LP or IP model. The LINDO
output file is in the DBC format which is an ASCII file. Those interested in the MPS
format may refer to IBM Mathematical Programming manuals for complete details. The
following simple example illustrates the MPS format input file and the DBC output file.

Normal format:

```
MIN  8 X - 2 Y - 3 Z
SUBJECT TO
  2)  X + 4 Y - Z >=  15
  3)  2 Y + 5 Z  <=  12
  4) - 4 X - 3 Y + 2 Z <=  24
END
```

MPS format:

```
NAME (MIN)
ROWS
 N 1
 G 2
 L 3
 L 4
```
LINDO output in DBC format:

-12.000000 1.0000000 F 0.10000000E+31
X 0.00000000E+00 8.0000000 C 0.10000000E+31
Y 6.0000000 0.00000000E+00 C 0.10000000E+31
Z 0.00000000E+00 2.0000000 C 0.10000000E+31

The first field in the first record of the output file is the final optimal objective value. The fields of the other records in this file are variable-name, variable-value, reduced-cost, variable-type, and simple-up-bound.

2.5 PROCESSING STEPS IN QCSMO

Step 1: Initialize the QCSMO scheduling environment. In this step, all the area related information is read in and saved to a QCSMO internal file called WORK.QP. The user's objectives are also converted to an internal data structure named OBJECT. LINDO is called and resides in memory. All relevant file handles are opened and ready for use.

Step 2: Build up QCSMO scheduling list, ARELIST. QCSMO reads from the WORK.QP file and finds the user specified starting areas then adds them to the ARELIST. If a LW area starts at the very beginning and its surrounding DV areas have not being mined, a pseudo LW area with zero mining tonnage (in programming, a positive but negligible amount of tonnage will be assigned to a pseudo LW area) will be inserted into
the ARELIST as a predecessor to the first LW. After the surrounding DV areas of the
first LW are completed, the pseudo LW will be dropped from the ARELIST and the first
LW begins mining.

Step 3: QCSMO checks if the ARELIST is empty. If yes, QCSMO scheduling
stops.

Step 4: Initialize the ARELIST at the beginning of each period. Based on the
assigned tonnage for each area in the list, QCSMO checks the average quality parameters
and corresponding advance in each area by interfacing with ICAMPS. Then QCSMO
determines the mining status of each area. When an area first appears in the list, that
area’s status is available (A). If mining is already in progress in the area, the status is
mining(M). If the area will be completely mined during the current period, the status is
done(D). If an LW area is in D status, QCSMO needs to find the next LW area and put it
in the list. If a DV area is completed, QCSMO will put all its possible successors into the
list for competitive selection by MLIP based upon quality and other considerations.

Step 5: QCSMO builds a MLIP model base on all the information stored in
ARELIST and saves the model in MPS format in a file named LNDIN.

Step 6: QCSMO interfaces with LINDO. LINDO solves the model and saves the
result in the LNDOUT file in the DBC format.

Step 7: QCSMO reads the LINDO output. Based on the result of MLIP and the
penalty coefficients provided by the user, QCSMO gets the assignment of tonnage and
calculates the cost of each quality deviation for each area. Finally, QCSMO saves all the
information into the GC file by area. QCSMO has completed one period work.

Step 8: QCSMO checks if any area has been completed. If yes, QCSMO updates
the mining status of the area in ARELIST and in WORK.QP file, so the whole system
knows the completion of this area. Then, the area is dropped out of ARELIST.

Step 9: QCSMO proceeds to next period.
CHAPTER III
BUILDING THE MLIP CONCEPTUAL MODEL

3.1 SYMBOLS AND TERMINOLOGY

QCSMO: Quality Constrained Scheduling of Mining Operations.
MLIP: Mixed Linear and Integer Programming.
LW: A Long Wall mining unit that mines long wall areas.
CM: Continuous Miner that mines development areas.
NDEV: Negative deviation from requirement.
PDEV: Positive deviation from requirement.

Conventions: In the MLIP model, all uppercase symbols represent unknowns and all lower case symbols represent known constants at the beginning of a given period. Usually, variables that have values from the previous period are in lower case. Variables that are unknown at the beginning of a period are in upper case because they will be assigned new values after LINDO solves the MLIP model. This convention does not apply to a subscript, which is used to indicate a specific period or mining unit. Sometimes the subscript of a symbol can be omitted, if the symbol is used for a general purpose.

3.2 VARIABLES DEFINITION

The QCSMO algorithm schedules a mine over a time horizon specified by the user. QCSMO dynamically builds an MLIP model according to specific mining conditions for each period within the horizon. Any changes of the scheduling requirements in one period only affect the following period(s).

The MLIP scheduling (Mixed LP and IP) model is built based on the following assumptions:

1. A mine has M Long Wall (LW) and N Continuous Miners (CM) mining units.
2. There are \( m \) LW crews and \( n \) CM crews. The total number of the crews is \( m + n \); \( m \leq M \) and \( n \leq N \).

3. For each scheduling period, the user specifies the total tonnage required from all LW units \( (T_{LW}) \) and CM units \( (T_{CM}) \). The user also specifies the production rate or capacity for each of the mining units in terms of tons per period; \( t_{LW} \) is the capacity of a LW unit and \( t_{CM} \) is the capacity of a CM unit. The scheduling algorithm itself does not use the number of crews explicitly. Instead, the number of crews is embedded implicitly in the tonnage requirements, that is, \( T_{LW} \) and \( T_{CM} \). If \( t_{LW} \) is the production rate per crew per period for each LW unit, then \( t_{LW} \) multiplied by the number of crews, or \( m \) in this case, working on LW units should yield \( T_{LW} \) which is the total tonnage required from the LW units in a time period. The same relation is true for the number of crews working on the CM units. Since the total tonnages required for the LW and CM units are known and the production rate for a particular period for each type of mining unit is also provided, the number of crews working in that period, either on LWs or on CMs, is implicitly determined by the tonnage requirement.

Theoretically, the production rate \( t_{LW} \) and \( t_{CM} \) can be arbitrary, but in QCSMO scheduling, there is a limitation for the upper bound of \( t_{LW} \) and \( t_{CM} \). If \( t_{LW} \) or \( t_{CM} \) is too large (or an area is too small), a mining unit could mine more than one area within a period. In such a case, an area could be started and completed within the same period. QCSMO would not be able to bring all its successors into scheduling, because QCSMO only checks mining conditions at the beginning of each period. In real mining, this limitation never causes any problem since it can be easily solved by merging the area that caused the problem to a larger neighboring area before scheduling and then dividing the production accordingly after scheduling.
4. K quality standards are also specified by the user and denoted by $Q_1, Q_2, Q_3, \ldots, Q_K$. The quality parameters together with the tonnage requirements determine a production vector defined as

$$P = (T_{Lw}, T_{CM}, Q_1, Q_2, Q_3, \ldots, Q_K).$$

The QCSMO algorithm uses the MLIP model to find out an optimal assignment of units to areas for each period. This assignment becomes part of the mining sequence for the entire time horizon. MLIP is also restricted by the available quality of coal and the mine layout. The quality of the coal varies within the coal seam and can only be estimated from samples drawn from the coal seam. Geological modeling tools provide an estimate of coal quality for rectangular blocks within the coal deposit.

The sequence of mining the coal is also restricted by the mine design or layout. Mining proceeds from certain starting points and only the areas that are adjacent to previously mined areas are accessible to the mining units. Other technological restrictions also apply so that the sequence in which areas can be mined in the future depends on past decisions. Figure 3.1 illustrates the precedence relation between a LW area and its surrounding DV areas.

The proposed mine layout and available coal resource may not allow MLIP to find a solution that satisfies the production vector $P$ in every period. If the production vector $P$ is satisfied in every period within the time horizon and the coal resources are exhausted, the mining sequence is said to be optimized. Hence, a global optimum has been reached. If $P$ cannot be satisfied for every period, the QCSMO scheduling algorithm will generate a mining sequence that guarantees minimizing the deviation from $P$ in each period. In the MLIP model, deviations from the specification are penalized. The penalties reflect the cost of meeting specification from external sources, such as purchasing coal from other mines for blending.
Note that the QCSMO algorithm does not optimize the mining sequence globally, instead it optimizes the sequence within each period, that is, the deviations from the production vector $P$ are minimized. Consequently, the QCSMO is ultimately a heuristic algorithm.

![Diagram showing the precedent relationship between a LW area and its surrounding DV areas.]

Figure 3.1 Precedent relationship between a LW area and its surrounding DV areas.

3.3 MLIP MODEL

The MLIP model is composed of three major parts: the objective function, the primary constraints and dynamic constraints. The following sections elaborate on this concept and the process of building these components.
3.3.1 MLIP OBJECTIVE FUNCTION

The general objective of the MLIP model is to minimize all possible deviations from the production vector \( P(T_{LW}, T_{CM}, Q_1, Q_2, Q_3, ..., Q_K) \). There are two major categories of deviation. One refers to production and the other refers to quality. Since the main purpose of mining is to produce coal, QCSMO treats the production requirement as a primary goal and gives it the highest priority. The quality is given a secondary priority though it is very important to this research. The relative priorities are reflected in the cost associated with deviating from the requirements.

The objective value of the MLIP model is interpreted as the cumulative loss in dollars caused by deviating from the given requirements. The deviation from a given requirement can be positive or negative; a positive deviation indicates an over or surplus production and vice versa. The heaviest weight is assigned to the most unwanted deviation. So the hierarchical importance of a requirement is indicated by the weight assigned to its corresponding positive or negative deviation. The weight of a deviation is representative of the true cost of that deviation.

The cost of the under production (negative deviation of production requirement) is the current market price for purchasing coal in the amount short. The under production incurs a very high penalty since it forces a coal producer to buy coal from its competitor to fill contracts. But the over production is not treated very seriously for it only adds some inventory and handling cost and perhaps some overtime paid to workers.

In the consideration of the quality requirements, the costs of the positive deviations for sulfur, ash and moisture and a negative deviation for B.T.U. are tangible and are treated as true losses, since the coal production is below the specified grade and would possibly reduce the price received for the coal. For example, the required sulfur content for a period may be two percent, but the actual content is three percent. Under the current market conditions, coal with two percent sulfur costs $5 more than coal with three
percent sulfur per ton. So the penalty or weight in the MLIP model could be set to $5 per ton for one percent sulfur content surplus.

On the other hand, the cost of the negative deviations for sulfur, ash and moisture and positive deviation for B.T.U. is intangible and is treated as a loss of opportunity cost, since the coal produced is a higher grade but the producer usually will not receive price compensation for supplying better than required coal to a customer. So the cost of better than required production is treated as a future loss of opportunity to sell the coal at a higher price to another customer.

The determination of the weights for all those deviations in the objective function is quite subjective, but some rules apply. In the practical application of this algorithm, the user usually can determine the correct weights because the coal producer is knowledgeable of the market. But, for the purpose of this research, the author weighted deviations based on the prices of the current coal market. Arbitrary values were assigned to those deviations with intangible opportunity cost.

### 3.3.2 MLIP PRIMARY CONSTRAINTS

Let $\text{LWT}_m (m = 1,...,M)$ be the tonnage assigned to each LW unit and $\text{CMT}_n (n = 1,...,N)$ be the tonnage assigned to each CM unit, the constraints are

\[
\text{LWT}_1 + \text{LWT}_2 + \cdots + \text{LWT}_M + \text{NDEV}_{\text{LW}} - \text{PDEV}_{\text{LW}} = T_M
\]

\[
\text{CMT}_1 + \text{CMT}_2 + \cdots + \text{CMT}_N + \text{NDEV}_{\text{CM}} - \text{PDEV}_{\text{CM}} = T_N
\]

$T_M$ and $T_N$ are the production requirements for the LW units and CM units respectively. $\text{NDEV}$ and $\text{PDEV}$ are variables that represent the positive and negative deviations from the specified requirements.

Production is constrained to the specified machine capacity by the following constraints:

\[
\text{LWT}_m \leq t_{\text{LW}} (m = 1,\ldots,M)
\]
Let $qcm_{nk}$ or $qlw_{mk}$ be the $k$th quality parameter representing the average coal quality of the DV areas or LW areas from which LWT or CMT tons of coal are produced. A group of constraints,

$$
(qlw_{11} - Q_1) * LWT_1 + (qlw_{12} - Q_2) * LWT_2 + \cdots + (qlw_{M1} - Q_1) * LWT_M
+ (qc_{m1} - Q_1) * CMT_1 + (qc_{m2} - Q_1) * CMT_2 + \cdots + (qc_{MN} - Q_1) * CMT_N
+ NDEV_1 - PDEV_1 = 0
$$

$$
(qlw_{12} - Q_2) * LWT_1 + (qlw_{22} - Q_2) * LWT_2 + \cdots + (qlw_{M2} - Q_2) * LWT_M
+ (qc_{m1} - Q_2) * CMT_1 + (qc_{m2} - Q_2) * CMT_2 + \cdots + (qc_{MN} - Q_2) * CMT_N
+ NDEV_2 - PDEV_2 = 0
$$

$$
(qlw_{1K} - Q_K) * LWT_1 + (qlw_{2K} - Q_K) * LWT_2 + \cdots + (qlw_{MK} - Q_K) * LWT_M
+ (qc_{m1} - Q_K) * CMT_1 + (qc_{m2} - Q_K) * CMT_2 + \cdots + (qc_{MN} - Q_K) * CMT_N
+ NDEV_K - PDEV_K = 0
$$

hold for each of the production quality requirements. Typically, the most critical requirements are the sulfur, moisture and ash content percentages and the energy content in B.T.U. per unit weight. The above constraints are required in the MLIP model for every scheduling period, so they are called the primary constraints.

### 3.3.3 MLIP Dynamic Constraints

The MLIP model is a dynamic model that is updated in every period to fit the new mining situation. There are four other sets of constraints included in the MLIP model. They are used to monitor each mining unit to determine whether each mining unit is mining or has completed a mining area. LW areas and CM areas are treated differently according to specific mining considerations.
Constraints for a LW unit are much more complicated than for a CM unit. This is because each LW unit has an extra scheduling requirement, i.e., under any circumstances, a LW unit should not be delayed, except when no LW area is available at the beginning of scheduling or when all LW areas have been mined.

LW units are far more productive than CM units and contribute the major part of the total production. Average LW cost per unit production is generally much less than for CM production. Any delay of a LW unit should be avoided. Otherwise, an idle LW unit would waste not only the investment in the LW unit but also the investment in all its supporting peripheral equipment. In addition, under production from the LW units can cause other problems such as failing to meet the total production quota.

According to the mining method, a LW area can be mined only after all its surrounding DV areas are completely mined. See Figure 3.2 for an example. While a LW area is currently being mined, the scheduling algorithm should be able to "look ahead" to make sure that the CM units, which are developing the DV areas for the next LW area, are keeping pace with the advance of the LW unit. The look-ahead feature ensures that the LW area one can be started as soon as the current LW area is completed.

Figure 3.2 Mining Sequence Requirements.
In order to make sure that no LW unit is delayed, the algorithm uses a group of constraints to compare the advance of each LW unit with the advances of the CM units working on the DV areas surrounding the next LW area. This is done by comparing the remaining tonnage in the current LW area and the remaining tonnage of all DV areas surrounding its successor. Figure 3.3 is an example where mining of a LW area and the DV areas for the next LW area are in progress. If the CM units are falling behind, the look-ahead constraints in the MLIP will force all those CM units to mine at full capacity until the proper ratio is restored between the remaining tonnage of the current LW area and the DV areas for the next area.

![Diagram showing remaining tonnage relationship of LW1 and the DV areas of LW2.](image)

Figure 3.3 Remaining tonnage relationship of LW1 and the DV areas of LW2.

The look-ahead mechanism was introduced into the MLIP model for the purpose of monitoring and maintaining the correct ratio of the remaining tonnage to mine out the current LW area to the remaining tonnage to mine out the DV areas that surround the next LW. The look-ahead constraints always stay active to ensure that the next LW area is ready to start before the time when the current LW area is completed. The look-ahead
constraints in MLIP can only monitor one LW area at a time. In the period when transition happens, the current LW area is completed and the next LW area is started, there is no look-ahead constraints for the next LW area, but the constraints will be restored in the next period when the previous "next" LW area becomes the "current" LW area. This situation is one of the weak points of the present MLIP model and will be discussed in Chapter V.

The dynamic constraints requires additional notations as following. The symbols with capital letters refer to the end of the current period; low case letters apply to end of the previous period.

LWT or CMT is the tonnage mined from a LW or a CM area in the current period.

t_LW or t_CM is the tonnage required for a LW or a CM area in the current period.

RMLWT and rmlwt: remaining tonnage to be mined from the current LW area, measured as of the end of the current period.

RMDVT and rmdvt: remaining tonnage to be mined from all the DV areas that surround the next LW area measured as of the end of the current period.

p/q: where p and q are the production rates for the LW and CM units respectively. For instance, p/q equal four means a LW unit can mine four times as much tonnage as a CM unit.

SLK: A slack variable used in MLIP to keep the model always feasible. If this variable appears in the objective function, it will have very large penalty coefficient so that it will be driven to zero whenever the MLIP model is feasible. An unfeasible MLIP model could occur during the iterative scheduling process under certain circumstances when a LW unit is moved to a new location or the sizes of adjacent LW areas vary to much. This type of variable avoids halting the whole scheduling process if conditions do not allow a feasible solution in a specific period.
Case 1: A LW unit is mining a LW area.

Tonnage constraint:

A LW unit must not exceed its capacity.

\[ LWT \leq t_{LW} \]

Look-ahead constraints:

The tonnage mined by a LW unit plus the remaining tons to be mined as of the end of the current period must equal to the remaining tons at the end of the previous period.

\[ LWT + RMLWT = rmlwt \]

The tonnage mined by all the CM units that are currently developing the DV areas surrounding the next LW area plus the remaining tonnage to be mined from these areas as of the end of the current period must be equal to the remaining tonnage to be mined for all those DV areas at the end of the previous period.

\[ \sum CMT_{(Mining \ DV \ for \ next \ LW)} + RMDVT = rmdvt \]

The remaining tonnage for the DV areas surrounding the next LW, adjusted by the advance ratio, must always be less than the remaining tonnage of the current LW to prevent delaying a LW unit. The SLK here is used to keep the constraint feasible even if a LW unit is actually delayed under certain circumstance.

\[ \frac{p}{q} \cdot RMDVT - RMLWT - SLK \leq 0 \]

Case 2: A LW unit is completing the current LW area and the next LW area is introduced into scheduling.

Tonnage constraint:

Tonnage mined by the unit from the current LW area and the next LW area must not exceed its capacity.

\[ LWT + LWT_{next} \leq t_{LW} \]
Tonnage mined from the current LW area must not exceed the remaining tonnage available as of the end of the previous period.

\[ \text{LWT} \leq \text{rmlwt} \]

Trigger constraints:

The next LW can be mined only after the current one is completed. \( Y_1 \) is a 0-1 integer variable which will be set to 1 by LINDO when the current LW is completed. \( Y_1 \) must be 1 to start next LW area.

\[ LWT_{\text{next}} \leq Y_1(t_{\text{LW}} - \text{rmlwt}) \]

\( Y_2 \) is set to 0 only when LWT equals rmlwt. BIG_M is a large number.

\[ \text{LWT} + \text{BIG}_M \times Y_2 \geq \text{rmlwt} \]

Both \( Y_1 \) and \( Y_2 \) cannot be equal to one.

\[ Y_1 + Y_2 = 1 \]

Look-ahead constraints:

The tonnage mined by a LW unit plus the remaining tonnage at the end of current period must be equal to the remaining tons at the end of the previous period.

\[ \text{LWT} + \text{RMLWT} = \text{rmlwt} \]

For the CM units that are currently developing the DV areas that surround the next LW area, the tonnage mined in this period plus the remaining tonnage at the end of the current period must equal to the remaining tonnage at the end of the previous period.

\[ \sum \text{CMT}(\text{for next LW}) + \text{RMDVT} = \text{rmdvt} \]

In order to prevent delaying a LW unit, the development of the DV areas, which surround the next LW area, should always keep pace with the mining advance of the current LW. If the next LW area can be started on time, then RMDVT, RMLWT, SLK are all set to zero.

\[ \frac{p}{q} \times \text{RMDVT} - \text{RMLWT} - \text{SLK} \leq 0 \]
Case 3: A CM unit is mining a DV area.

Tonnage constraint:

A CM unit must not exceed its tonnage capacity.

\[ CMT \leq t_{CM} \]

Case 4: A CM unit is completing a DV area.

When a CM unit compete a DV area, it can proceed to mine one, two or three adjacent DV areas as shown in Figure 3.4

![Figure 3.4 Possible CM Mining Options.](image)

All its possible successors are put into a competitive selecting process so that the coal quality can be adjusted by choosing the correct area and proper advance in the selected area. If an area is not selected the first time, it will compete for the next selection.

Tonnage constraint:

Tonnage mined in the current DV area must not exceed the remaining tonnage at the end of the previous. Here rmdvt means the remaining tonnage of an individual DV area at the end of previous period.

\[ CMT \leq rmdvt \]
Trigger constraints:

A CM unit cannot begin mining until the current one is completed. $Y_1$ is a 0-1 integer variable that will be set to 1 when the current DV is completed.

$$CMT_{next1} \leq Y_1 \times (t_{CM} - rmdvt)$$

$$CMT_{next2} \leq Y_1 \times (t_{CM} - rmdvt)$$

$$\ldots$$

$$CMT_{nexti} \leq Y_1 \times (t_{CM} - rmdvt)$$

Only when $CMT$ equals rmdvt, is $Y_2$ set to 0.

$$CMT + BIG_M \times Y_2 \geq rmdvt$$

Both $Y_1$ and $Y_2$ cannot be one, that is

$$Y_1 + Y_2 = 1$$

3.4 Summary of MLIP Dynamic Constraints

For easy referencing, all the dynamic constraints are listed according to the mining conditions as follows:

Case 1: A LW unit is mining a LW area.

$$LWT \leq t_{LW}$$

$$LWT + RMLWT = rmlwt$$

$$\sum CMT (Mining DV for next LW) + RMDVT = rmdvt$$

$$p/q * RMDVT - RMLWT - SLK \leq 0$$

Case 2: A LW unit is completing the current LW area and the next LW area is introduced into scheduling.

$$LWT + LWT_{next} \leq t_{LW}$$

$$LWT \leq rmlwt$$

$$LWT_{next} \leq Y_1 \times (t_{LW} - rmlwt)$$

$$LWT + BIG_M \times Y_2 \geq rmlwt$$
\[ Y_1 + Y_2 = 1 \]
\[ \text{LWT} + \text{RMLWT} = \text{rmlwt} \]
\[ \sum \text{CMT}_{\text{for next LW}} + \text{RMDVT} = \text{rmdvt} \]
\[ \frac{p}{q} \times \text{RMDVT} - \text{RMLWT} - \text{SLK} \leq 0 \]

**Case 3: A CM unit is mining a DV area.**

\[ \text{CMT} \leq t_{cm} \]

**Case 4: A CM unit is completing a DV area.**

\[ \text{CMT} \leq \text{rmdvt} \]

\[ \text{CMT}_{\text{next1}} \leq Y_1 \times (t_{CM} - \text{rmdvt}) \]
\[ \text{CMT}_{\text{next2}} \leq Y_1 \times (t_{CM} - \text{rmdvt}) \]
\[ \ldots \]
\[ \text{CMT}_{\text{nexti}} \leq Y_1 \times (t_{CM} - \text{rmdvt}) \]

\[ \text{CMT} + \text{BIG}_M \times Y_2 \geq \text{rmdvt} \]

\[ Y_1 + Y_2 = 1 \]

Note that in the above four cases, whenever a new area is introduced into the MLIP, all the primary constraints and the objective function should reflect the presence of this new area by inserting a corresponding variable that represents the tonnage assigned to that area in those constraints.
CHAPTER IV

A CASE STUDY BY APPLYING QCSMO

4.1 CASE DESCRIPTION

The Quality Constrained Scheduling of Mining Operations (QCSMO) algorithm is tested on a real mine layout of the XYZ Coal Co. as shown in Figure 4.1.

![Figure 4.1 Mine Layout of XYZ Coal Co.](image)

The user has provided the following information for QCSMO.

1. The Area Definition File named WORK.SE, which contains the information about each area that is going to be scheduled, such as Identification number (ID), coordinates, and other technical information.

2. The grid file named MINE3.GRD, which contains all the geological information about that mine.

3. The area relation file named RELATION, which contains the mining sequence of
LW areas and precedence relationship of LW and DV areas.

4. The objective definition file OBJECT.DEF, in which the user defines the mining unit capacity, specifies the mining production and quality requirements and the cost of any deviation from specifications.

4.2 INITIALIZING QCSMO SCHEDULING

Based on the WORK.SE and RELATION files, a QCSMO internal file named WORK.QP (Quality Planning) was created. Original WORK.QP is in the BTree format. Part of the of WORK.QP is shown below in ASCII format.

<table>
<thead>
<tr>
<th>AreaID</th>
<th>Type</th>
<th>Status</th>
<th>Tons</th>
<th>Adv</th>
<th>Sulfur</th>
<th>Ash</th>
<th>Pext</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000133</td>
<td>L</td>
<td>U</td>
<td>663246.98</td>
<td>3844.91</td>
<td>3.28</td>
<td>7.45</td>
<td>100.00</td>
</tr>
<tr>
<td>0000143</td>
<td>L</td>
<td>U</td>
<td>852996.97</td>
<td>4944.91</td>
<td>3.27</td>
<td>7.44</td>
<td>100.00</td>
</tr>
<tr>
<td>0000153</td>
<td>L</td>
<td>U</td>
<td>1871605.83</td>
<td>9764.90</td>
<td>3.38</td>
<td>7.45</td>
<td>100.00</td>
</tr>
<tr>
<td>0000462</td>
<td>C</td>
<td>U</td>
<td>83486.56</td>
<td>4385.00</td>
<td>3.27</td>
<td>7.50</td>
<td>38.21</td>
</tr>
<tr>
<td>0000472</td>
<td>C</td>
<td>U</td>
<td>28828.43</td>
<td>1095.00</td>
<td>3.30</td>
<td>6.99</td>
<td>40.40</td>
</tr>
<tr>
<td>0000492</td>
<td>C</td>
<td>U</td>
<td>104414.29</td>
<td>5485.00</td>
<td>3.31</td>
<td>7.50</td>
<td>38.20</td>
</tr>
<tr>
<td>0000502</td>
<td>C</td>
<td>U</td>
<td>126186.28</td>
<td>4815.00</td>
<td>3.35</td>
<td>6.88</td>
<td>40.40</td>
</tr>
<tr>
<td>0000522</td>
<td>C</td>
<td>U</td>
<td>196198.17</td>
<td>10305.00</td>
<td>3.48</td>
<td>7.55</td>
<td>38.21</td>
</tr>
<tr>
<td>0000562</td>
<td>C</td>
<td>U</td>
<td>286139.17</td>
<td>13565.00</td>
<td>4.23</td>
<td>7.94</td>
<td>42.33</td>
</tr>
<tr>
<td>0000592</td>
<td>C</td>
<td>U</td>
<td>34164.05</td>
<td>1235.00</td>
<td>-99.00</td>
<td>-99.00</td>
<td>42.84</td>
</tr>
<tr>
<td>0100302</td>
<td>C</td>
<td>A</td>
<td>52240.35</td>
<td>1160.00</td>
<td>3.64</td>
<td>8.36</td>
<td>38.21</td>
</tr>
<tr>
<td>0200302</td>
<td>C</td>
<td>U</td>
<td>52240.35</td>
<td>1160.00</td>
<td>3.64</td>
<td>8.36</td>
<td>38.21</td>
</tr>
<tr>
<td>0200442</td>
<td>C</td>
<td>U</td>
<td>94457.83</td>
<td>3535.00</td>
<td>3.30</td>
<td>7.28</td>
<td>41.00</td>
</tr>
<tr>
<td>0200452</td>
<td>C</td>
<td>U</td>
<td>11847.30</td>
<td>450.00</td>
<td>3.29</td>
<td>7.05</td>
<td>40.40</td>
</tr>
<tr>
<td>0200482</td>
<td>C</td>
<td>U</td>
<td>11847.30</td>
<td>450.00</td>
<td>3.32</td>
<td>6.95</td>
<td>40.40</td>
</tr>
<tr>
<td>0200512</td>
<td>C</td>
<td>U</td>
<td>13265.62</td>
<td>500.00</td>
<td>-99.00</td>
<td>-99.00</td>
<td>40.71</td>
</tr>
<tr>
<td>0199993</td>
<td>L</td>
<td>A</td>
<td>10.00</td>
<td>0.00</td>
<td>3.00</td>
<td>8.00</td>
<td>100.00</td>
</tr>
<tr>
<td>0300302</td>
<td>C</td>
<td>U</td>
<td>52240.23</td>
<td>1160.00</td>
<td>3.53</td>
<td>8.51</td>
<td>38.21</td>
</tr>
<tr>
<td>0400302</td>
<td>C</td>
<td>U</td>
<td>56743.57</td>
<td>1260.00</td>
<td>3.65</td>
<td>8.67</td>
<td>38.21</td>
</tr>
<tr>
<td>0600302</td>
<td>C</td>
<td>A</td>
<td>61246.72</td>
<td>1360.00</td>
<td>4.67</td>
<td>9.00</td>
<td>38.21</td>
</tr>
</tbody>
</table>

The meanings of each column are:

AreaID is a 7 digits(d1d2d3d4d5d6d7) Identification number, in which d1d2 are used to represent a subdivided area, d3d4d5d6 are main area code, d7 is the area type. If the type is 3, it means a LW area. Here a special long wall area 0199993 called a pseudo LW is inserted before the first real long wall area.
Type means Area type; L refers to LW, C refers to DV.

Status means Mining status. Here, A means Available; U means Unavailable; M means Mining; W means waiting and D means done. At the beginning of scheduling, only starting areas are marked A; others are marked U.

Tons means total tons of coal that can be mined in an area.

Adv means total length of advance within an area.

Sulfur means sulfur content in percentage per unit weight. If no information available, -99.00 appears in the field and the average value for that parameter will be assigned for scheduling purpose.

Ash means the ash content in percentage per unit weight.

Pext means percentage of extraction. In LW areas one hundred percent of the coal is extracted. In DV areas, the percentage depends upon how much coal is left for pillars.

The RELATION file shown below presents the area spatial precedence relation. The data has been reorganized for easier understanding. Here AreaID denotes an area. NextLW is the next LW area ID. SurrDV is the area ID for one of the surrounding DV areas. NextDV is the DV area that can be mined after this DV area. 0000000 indicates an area that has no pre-specified follower area.

LW area relation:

<table>
<thead>
<tr>
<th>AreaID</th>
<th>NextLW</th>
<th>SurrDV</th>
<th>SurrDV</th>
<th>SurrDV</th>
<th>SurrDV</th>
<th>SurrDV</th>
<th>SurrDV</th>
<th>SurrDV</th>
<th>SurrDV</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000133 0000143 0100302 0200302 0100442 0200442 0100452 0200452 0000462 0000472</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00000143 0000153 0000462 0300302 0000472 0100482 0200482 0000492 0000502</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000153 0000000 0400302 0000492 0100512 0200512 0000502 0000522</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

DV area relation:

<table>
<thead>
<tr>
<th>AreaID</th>
<th>NextDV</th>
<th>NextDV</th>
</tr>
</thead>
<tbody>
<tr>
<td>0100302 0200302 0100442</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0200302 0300302 0000462</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100442 0200442</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0200442 0100452</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000462 0200452 0000472</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000472 0100482</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0300302 0400302 0000492</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000492 0200482 0000502</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000502 0100512</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For instance, LW area 0000133 is followed by LW 0000143 and has surrounding DV areas 0100302, 0200302, 0100442, 0200442, 0100452, 0200452, 0000462, 0000472. DV area 0100302 is followed by two DV areas 0300302 and 0100402. This information is clearly shown in the mine layout.

The scheduling objective file OBJECT.DEF is shown below. The data is also reorganized to be meaningful.

Starting areas:
0000133 0100302 0600302

Production Requirement (tons per period):
LW: 50,000 tons DV: 20,000 tons

Capacity of Mining units (tons per period):
LW: 50,000 tons DV: 10,000 tons

Quality requirements (percentage per unit weight):
sulfur 3.0 ash 8.0 moisture 0 BTU 0

Cost of quality deviations ($ for +1% per unit weight)
positive deviation
sulfur 1.0 ash 1.0 moisture 2 BTU 0.5
negative deviation
sulfur 2.0 ash 0.5 moisture 2 BTU 0.5

Cost of production deviations ($ for +1 ton)
positive deviation
LW 3 DV 5
negative deviation
LW 25 DV 25

The data file indicates that the mining can start from area 0000133, 0100302 and 0600302. Since the DV areas surrounding LW area 0000133 have not been mined when scheduling begins, a pseudo LW will be inserted before LW 0000133. The function of a pseudo LW is to monitor the mining of all those DV areas that surrounding LW 0000133. Once all the DV areas are mined, the pseudo LW will start the real LW immediately. The total production required from LW and CM units implies that there are three crews, one
for the LW and one for each CM mining units.

A quality requirement of zero means there is no requirement for it. The quality requirements show three percent per unit weight for sulfur and eight percent for ash. The cost in dollars for quality deviations is based on one percent deviation per unit weight. For example, it will cost the XYZ company $1 for each one percent sulfur content over 3.0 and $2 for one percent under. The cost in dollars for production deviation is base on tons for over or under production. For LW units, $25 per ton for under production is showing the cost of buying coal from an outside source and $3 per ton for over production implies an extra cost for inventory and handling.

Before starting to schedule, QCSMO creates a linked list named ARELIST, which contains all areas that will be scheduled based on the information in WORK.QP. Each node in the ARELIST is an area record in which all the required information to schedule that area is stored. Then a MLIP model, which reflects the mining conditions in the current period, is built by checking the ARELIST for all the information needed. Areas are mined while strictly adhering to the precedence relationships. This procedure ensures that the result of QCSMO fits the real world. At the very beginning, the ARELIST always contains the starting areas assigned by the user. When an area is completed, QCSMO to brings all the next area(s) to be mined into the ARELIST and it drops the completed area(s) from the ARELIST. The QCSMO scheduling stops when the ARELIST becomes empty.

4.3 SCHEDULE PERIOD BY PERIOD

After initialization, QCSMO has built an ARELIST, which contains all the areas to be scheduled. The next step is to create a MLIP model for LINDO to solve. The MLIP model is dynamically created for each period according to the mining conditions at the beginning of that period. Building a MLIP model in MPS format is a complicated
procedure and is done by a routine called WT_LINDO. After LINDO solves the MLIP model, the solution is stored in a file named LNDOUT that is read by a routine called RD_LINDO. QCSMO then records the resulting schedule in the GC file and updates the ARELIST according to the new mining conditions. QCSMO updates the scheduling list by dropping any area from the ARELIST when it is completed and by inserting any area into it when its predecessors are completed. After updating the ARELIST, QCSMO is ready to start a new period or it stops if ARELIST is empty.

All the models shown below were manually converted from MPS format to normal mathematical format for easier understanding.

The total number of MLIP models or the periods required to schedule all the areas varies according to the production rate per period. The larger the production rate, the fewer the number of periods needed for scheduling. As discussed in the section 3.2 of Chapter III, the QCSMO cannot accept too large a production rate, since it would cause an area to be started and completed within the same period. In the example, there are 100 periods all together.

The following MLIP models were selected to show some critical stages of the QCSMO scheduling. A model and its solution are provided together with a short description about the process.

Some variables used in the following MLIP models are explained again for easier understanding. PDEV and NDEV denote positive and negative deviations respectively. The meaning of them depends on the constraint in which they appear. The symbol LW followed by a number indicates a LW area. The symbol DV followed by a number denotes a DV area. SLK is a slack variable and more than one SLK may exist in a MLIP model. Also RMLW and RMDV denote the remaining tonnage for a LW area and all the DV areas surrounding the next LW area respectively at the end of each period. All the Ys are integer variables and they are used to monitor the precedence and look-ahead
relationships. Also, notice the naming and numbering of constraints. If a constraint is not
named, it is assigned a number by LINDO.

**MODEL1: (Beginning of Scheduling)**

```
MIN PDEV1 + 2 NDEV1 + PDEV2 + 0.5 NDEV2 + 3 PDEV5
     + 25 NDEV5 + 5 PDEV6 + 25 NDEV6 + 9999999 SLK1
SUBJECT TO
SULF) - PDEV1 + NDEV1 + 0.78 DV30_1 + 1.39 DV30_6
     + 0.25 LW13 = 0
ASH) - PDEV2 + NDEV2 + 0.23 DV30_1 + 0.93 DV30_6
     - 0.93 LW13 = 0
TONL) - PDEV5 + NDEV5 + LW9999_1 + LW13 = 50000
TONC) - PDEV6 + NDEV6 + DV30_1 + DV30_6 = 20000
6) DV30_1 <= 10000
7) LW9999_1 <= 10
8) 9999999 Y1 + LW9999_1 >= 10
9) Y1 + Y2 <= 1
10) LW9999_1 + LW13 <= 50000
11) -49990 Y2 + LW13 <= 0
12) LW9999_1 + RMLW1 = 10
13) DV30_1 + RMDV1 = 316228.8
14) - RMLW1 + 4 RMDV1 - SLK1 <= 0
15) DV30_6 <= 10000
END
SUB Y1 1.00000
INTE Y1
SUB Y2 1.00000
INTE Y2

OBJECTIVE FUNCTION VALUE
1) 0.1224905E+14
VARIABLE VALUE REDUCED COST
Y1 1.000000 0.000000
Y2 0.000000 -1283295.500000
PDEV1 21654.880859 0.000000
NDEV1 0.000000 3.000000
PDEV2 11612.490234 0.000000
NDEV2 0.000000 1.500000
PDEV5 0.000000 28.000000
NDEV5 50000.000000 0.000000
PDEV6 0.000000 30.000000
NDEV6 0.000000 0.000000
DV30_1 10000.000000 0.000000
LW9999_1 0.000000 9999974.000000
DV30_6 10000.000000 0.000000
LW13 0.000000 0.000000
RMLW1 10.000000 0.000000
RMDV1 306228.750000 0.000000
SLK1 1224905.000000 0.000000
```

Description:

A pseudo long wall area LW9999_1 was inserted before the first long wall area
(LW13) to ensure that the development areas which are surrounding it are completely 
mined before it is scheduled. The pseudo long wall area is arbitrarily assigned a negligible 
amount of production. In this case, 10 tons were used. Constraints 7), 8), 9), 10) and 11) 
are used to trigger mining of the first real long wall area LW13 when the value of slack 
variable SLK1 is set to zero by the look-ahead constraints 12), 13) and 14). The total cost 
is very high in this case since the long wall unit is not in production.

MODEL 2 (A DV area is completed)

\[
\begin{align*}
\text{MIN} \quad & PDEV1 + 2 NDEV1 + PDEV2 + 0.5 NDEV2 + 3 PDEV5 \\
& + 25 NDEV5 + 5 PDEV6 + 25 NDEV6 + 9999999 SLK1 \\
\text{SUBJECT TO} \\
SULF) - PDEV1 + NDEV1 + 0.73 DV30_1 + 1.5 DV30_6 \\
& + 0.25 LW13 + 0.73 DV30_2 + 0.64 DV44_1 = 0 \\
ASH) - PDEV2 + NDEV2 + 0.28 DV30_1 + 0.95 DV30_6 \\
& - 0.93 LW13 + 0.29 DV30_2 + 0.03 DV44_1 = 0 \\
TONL) - PDEV5 + NDEV5 + LW99999_1 + LW13 = 50000 \\
TONC) - PDEV6 + NDEV6 + DV30_1 + DV30_6 + DV30_2 + DV44_1 \\
& = 20000 \\
& 6) \quad DV30_1 = 808. \\
& 7) \quad 9999999 Y1 + DV30_1 = 808. \\
& 8) \quad Y1 + Y2 = 1 \\
& 9) - 9192 Y2 + DV30_2 = 0 \\
& 10) - 9192 Y2 + DV44_1 = 0 \\
& 11) \quad LW99999_1 = 10 \\
& 12) \quad LW99999_1 + LW99999_1 = 10 \\
& 13) \quad Y3 + Y4 = 1 \\
& 14) \quad LW99999_1 + LW13 = 50000 \\
& 15) - 49990 Y4 + LW13 = 0 \\
& 16) \quad LW99999_1 + RMLW1 = 10 \\
& 17) \quad DV30_1 + DV30_2 + DV44_1 + RMDV1 = 306228. \\
& 18) - RMLW1 + 4 RMDV1 - SLK1 = 0 \\
& 19) \quad DV30_6 = 10000 \\
\end{align*}
\]

END

SUB \hspace{1cm} Y1 \hspace{1cm} 1.00000
INTE \hspace{1cm} Y2 \hspace{1cm} 1.00000
SUB \hspace{1cm} Y3 \hspace{1cm} 1.00000
INTE \hspace{1cm} Y4 \hspace{1cm} 1.00000

OBJECTIVE FUNCTION VALUE

\[ \text{1) \quad 0.1148139E+14} \]

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>VALUE</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y1</td>
<td>0.0000000</td>
<td>0.0000000</td>
</tr>
<tr>
<td>Y2</td>
<td>1.0000000</td>
<td>*</td>
</tr>
<tr>
<td>Y3</td>
<td>1.0000000</td>
<td>0.0000000</td>
</tr>
</tbody>
</table>
Description:

Constraints 6), 7), 8) and 9) are used to introduce the successors of DV30_1, DV30_2 and DV44_1, into the competitive selection. Note that the tonnage mined by DV30_2 and DV44_1 indicates that mining began in both areas immediately after DV30_1 was completed.

Model 3 (A LW area is completed)

MIN PDEV1 + 2 NDEV1 + PDEV2 + 0.5 NDEV2 + 3 PDEV5 + 25 NDEV5 + 5 PDEV6 + 25 NDEV6 + 9999999 SLK1

SUBJECT TO

SULF) - PDEV1 + NDEV1 + 1.69 DV30_6 + 0.51 LW13 + 0.30 LW14 = 0

ASH) - PDEV2 + NDEV2 + 1.0 DV30_6 + 0.08 LW13 - 1.02 LW14 = 0

TONL) - PDEV5 + NDEV5 + LW13 + LW14 = 50000

TONC) - PDEV6 + NDEV6 + DV30_6 + DV52 + DV51_1 = 20000

6) DV30_6 <= 10000

7) LW13 <= 14481.18

8) 9999999 Y1 + LW13 >= 14481.18

9) Y1 + Y2 <= 1

10) LW13 + LW14 <= 50000

11) - 35518.82 Y2 + LW14 <= 0

12) LW13 + RMLW1 = 14481.18

13) RMDV1 = 0

14) - RMLW1 + 4 RMDV1 - SLK1 <= 0

15) DV52 <= 10000

16) DV51_1 <= 10000

END

SUB Y1 1.00000
INTE Y1
SUB Y2 1.00000
INTE Y2

<table>
<thead>
<tr>
<th>Y4</th>
<th>0.00000</th>
<th>-1283295.50000</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDEV1</td>
<td>14430.522461</td>
<td>0.000000</td>
</tr>
<tr>
<td>NDEV1</td>
<td>0.000000</td>
<td>3.000000</td>
</tr>
<tr>
<td>PDEV2</td>
<td>3939.172852</td>
<td>0.000000</td>
</tr>
<tr>
<td>NDEV2</td>
<td>0.000000</td>
<td>1.500000</td>
</tr>
<tr>
<td>PDEV5</td>
<td>0.000000</td>
<td>28.000000</td>
</tr>
<tr>
<td>NDEV5</td>
<td>50000.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>PDEV6</td>
<td>0.000000</td>
<td>7.455857</td>
</tr>
<tr>
<td>NDEV6</td>
<td>0.000000</td>
<td>22.544144</td>
</tr>
<tr>
<td>DV30_1</td>
<td>808.362915</td>
<td>0.000000</td>
</tr>
<tr>
<td>LW99999_1</td>
<td>0.000000</td>
<td>9999974.000000</td>
</tr>
<tr>
<td>DV30_6</td>
<td>808.363647</td>
<td>0.000000</td>
</tr>
<tr>
<td>LW13</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>DV30_2</td>
<td>9191.636719</td>
<td>0.000000</td>
</tr>
<tr>
<td>DV44_1</td>
<td>9191.636719</td>
<td>0.000000</td>
</tr>
<tr>
<td>RMLW1</td>
<td>10.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>RMDV1</td>
<td>287037.125000</td>
<td>0.000000</td>
</tr>
<tr>
<td>SLK1</td>
<td>1148138.500000</td>
<td>0.000000</td>
</tr>
</tbody>
</table>
### OBJECTIVE FUNCTION VALUE

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>VALUE</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y1</td>
<td>0.000000</td>
<td>859204.687500</td>
</tr>
<tr>
<td>Y2</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>PDEV1</td>
<td>18083.392578</td>
<td>0.000000</td>
</tr>
<tr>
<td>NDEV1</td>
<td>0.000000</td>
<td>3.000000</td>
</tr>
<tr>
<td>PDEV2</td>
<td>0.000000</td>
<td>1.500000</td>
</tr>
<tr>
<td>NDEV2</td>
<td>35089.066406</td>
<td>0.000000</td>
</tr>
<tr>
<td>PDEV5</td>
<td>0.000000</td>
<td>28.000000</td>
</tr>
<tr>
<td>NDEV5</td>
<td>0.001953</td>
<td>0.000000</td>
</tr>
<tr>
<td>PDEV6</td>
<td>0.000000</td>
<td>5.000000</td>
</tr>
<tr>
<td>NDEV6</td>
<td>0.000000</td>
<td>25.000000</td>
</tr>
<tr>
<td>DV30_6</td>
<td>0.000000</td>
<td>1.186535</td>
</tr>
<tr>
<td>LW13</td>
<td>14481.181641</td>
<td>0.000000</td>
</tr>
<tr>
<td>DV52</td>
<td>10000.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>LW14</td>
<td>35518.816406</td>
<td>0.000000</td>
</tr>
<tr>
<td>DV51_1</td>
<td>10000.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>RMLW1</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>RMDV1</td>
<td>0.000000</td>
<td>98.104515</td>
</tr>
<tr>
<td>SLK1</td>
<td>0.000000</td>
<td>9999974.000000</td>
</tr>
</tbody>
</table>

Description:

In this model, LW13 was completed and LW14 were started immediately after LW13. 7), 8), 9), 10), 11) and 12) are trigger constraints. 13), 14) and 15) are look-ahead constraints. Note that RMDV1 is already equal to zero in constraint 13). This indicates that there was actually no restriction on starting LW14, since all its surrounding DV areas have been mined.

**Model 4 (A LW unit is set idle by MLIP)**

\[
\begin{align*}
\text{MIN} & \quad \text{PDEV1} + 2 \text{NDEV1} + \text{PDEV2} + 0.5 \text{NDEV2} + 3 \text{PDEV5} + 25 \text{NDEV5} + 5 \text{PDEV6} + 25 \text{NDEV6} + 9999999 \text{SLK1} \\
\text{SUBJECT TO} & \\
\text{SULF}) & \quad \text{PDEV1} + \text{NDEV1} + 1.69 \text{DV30_6} + 0.24 \text{LW13} + 0.69 \text{DV30_4} + 0.27 \text{DV49} = 0 \\
\text{ASH}) & \quad \text{PDEV2} + \text{NDEV2} + 1.0 \text{DV30_6} - 0.90 \text{LW13} + 0.70 \text{DV30_4} - 0.43 \text{DV49} = 0 \\
\text{TONL}) & \quad \text{PDEV5} + \text{NDEV5} + \text{LW13} = 50000 \\
\text{TONC}) & \quad \text{PDEV6} + \text{NDEV6} + \text{DV30_6} + \text{DV30_4} + \text{DV49} = 20000 \\
6) & \quad \text{DV30_6} \leq 10000 \\
7) & \quad \text{LW13} \leq 50000 \\
8) & \quad \text{LW13} + \text{RMLW1} = 613257 \\
9) & \quad \text{DV49} + \text{RMDV1} = 206361.6 \\
10) & \quad \text{RMLW1} + 4 \text{RMDV1} - \text{SLK1} \leq 0 \\
11) & \quad \text{DV30_4} \leq 10000 \\
12) & \quad \text{DV49} \leq 10000 \\
\text{END}
\end{align*}
\]
In the solution, the LW13 was not assigned any work in this period. By comparing the right-hand sides of the look-ahead constraints 8) and 9), one can be seen that the remaining tonnage in LW13 is considerably smaller than the remaining tonnage of all the surrounding DV areas of LW14. This indicates that if LW13 had kept mining until it was completed, there would be no chance to start LW14, because the two CM units could not complete all the DV areas that surround LW14. The problem is caused by the mine layout. Because LW13 is much smaller than successor LW14, mining the surrounding DV areas of the large LW panel takes much longer than mining LW13. This solution shown here is reasonable; it illustrates the built-in policy of QCSMO scheduling, i.e., if a LW must be delayed, let the delay happen as early as possible. To do so will prevent unnecessary premature investment in LW production. The other option is to increase the production rate or production time for the development work and rerun QCSMO.

### 4.4 THE RESULT OF QCSMO SCHEDULING

The results of QCSMO scheduling are stored in an output file called GC. The GC file is composed of records which hold information about areas scheduled in each of the periods. Each period has a header record marking the start of a period and a tail record...
which indicates the end of a period. In between, all areas that are mined in this period will be recorded along with the tonnage mined, quality deviations, cost of deviation and other relevant information. A typical period record is shown below.

<table>
<thead>
<tr>
<th>Period</th>
<th>AreaID</th>
<th>Type</th>
<th>Tons1</th>
<th>Tons2</th>
<th>Sulfur</th>
<th>Ash</th>
<th>CostSulfur</th>
<th>CostAsh</th>
</tr>
</thead>
<tbody>
<tr>
<td>004500</td>
<td>start</td>
<td>S</td>
<td>50000</td>
<td>20000</td>
<td>3.00</td>
<td>8.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>004501</td>
<td>0000133</td>
<td>L</td>
<td>50000</td>
<td>14481</td>
<td>3.51</td>
<td>8.08</td>
<td>7426.72</td>
<td>1129.02</td>
</tr>
<tr>
<td>004502</td>
<td>0000522</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
<td>8.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>004503</td>
<td>0001433</td>
<td>L</td>
<td>50000</td>
<td>35519</td>
<td>3.30</td>
<td>6.98</td>
<td>10656.67</td>
<td>18109.03</td>
</tr>
<tr>
<td>004504</td>
<td>0100512</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
<td>8.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>004505</td>
<td>ending</td>
<td>E</td>
<td>50000</td>
<td>20000</td>
<td>0.26</td>
<td>-0.50</td>
<td>18083.39</td>
<td>17544.52</td>
</tr>
</tbody>
</table>

The header record is explained as follows: 004500 is the starting record of the 45th period; AreaID and type are used to mark the start of a period; Tons1 and Tons2 are the total production required in this period from the LW and CM units respectively. Sulfur and Ash are the quality specifications in percentage per unit weight; CostSulfur and CostAsh are not used.

The tail record is explained as follows: 004505 is the end record of the 45th period; AreaID and type are used to mark the end of a period; Tons1 and Tons2 are the actual production in this period from the LW and CM units respectively. Sulfur and Ash are quality deviations from the specifications. The units are percentage per unit weight. CostSulfur and CostAsh indicates the total cost in dollars caused by the deviations in that period. For example, the CostSulfur shows 18083.39 dollars loss for 70,000 tons total production caused by an excessive sulfur content of 0.26 percent. The CostAsh shows that XYZ Coal CO. will lose $17544.52 in opportunity cost because the ash content is 0.5 less than required.

For the records in between, AreaID is the Identification number for the mined area and Type is the area type; Tons1 is the capacity of the mining unit that has worked in that area. Tons2 is the actual tonnage yield from that area. Sulfur and Ash are actual quality parameters for the portion of the area mined in the 45th period. CostSulfur and CostAsh are costs for quality deviations from that area.
The most important feature of the GC file is that it contains chronological records of all the mining activities throughout the time horizon. It should be mentioned again that no precedence was violated for the mining sequence contained in the GC file, so the mining management can use the mining sequence from GC file to make a practical mining plan or tailor it for other operational management purposes.

Since the full GC file for this example is very lengthy, it is listed separately in APPENDIX B.
CHAPTER V
CONCLUSIONS

5.1 FURTHER DISCUSSION ON QCSMO

As shown in the case study of Chapter IV, QCSMO was tested with data from a mine which has one LW and two CM mining units. QCSMO was also expanded to include a variety of initial mining conditions, in which a mine can have zero or one LW units and two or more CM units. In the no long walls case, the CM units are no longer forced to perform the development work for LW areas, and can mine according to the precedence restrictions and quality requirements. If QCSMO scheduling is not restricted by LW operations, the results more closely resemble an underground blending process, because CM units can mine the available DV areas selectively to meet the production and quality requirements.

Theoretically, there is no limitation on how many LW and CM units can work in the system. In the real world, however, a mine usually has no more than 20 mining units. Therefore, in the QPLAN.H file the variable MXU is defined as 20 to limit the maximum number of mining units. The users, however, can change the MXU in QPLAN.H to any number they want.

To function properly, QCSMO requires a minimum hardware configuration. The author used an IBM compatible 486 DX-33 with 4MB RAM and 120MB hard drive system to develop this algorithm. It is highly recommended that QCSMO be used in a system that is at least comparable to the above configurations.

The major objective of QCSMO, as discussed in Chapter I, was to generate a mining production sequence; that is, when and where a mining unit should work and how much production should come from the assigned area(s) subject to the quantity and quality constraints.
Like any other heuristic methods, it is very difficult to verify the optimality of QCSMO schedules. However, if the user is very careful in setting of the model, QCSMO will achieve a better solution. The following are suggested.

First, the mining sequence for the DV areas that surround LW areas is partially determined by the LW mining sequence. Since QCSMO follows a predetermined sequence to mine the LW areas, and the total number of LW mining sequences is usually very limited, there is a great possibility to find the optimal solution by trying all possible LW sequences and comparing the results.

Second, the QCSMO schedules a mine by areas and the layout of areas, such as, mining directions, size of areas or spatial relation among areas, has great impact on the final solution. Varying the mining layout by trial and error will give QCSMO a better chance to find the optimal solution.

Third, the most important and attractive feature of QCSMO is its flexibility. QCSMO could also be used in two extreme cases for special planning purposes. Since a QCSMO solution is always feasible, a user can use this feature to find a workable mining sequence by assigning zero penalty to each quality deviation. A user can also generate a mining sequence for best-quality-fitting by assigning zero weights to the production deviations. The latter sequence allows the management to evaluate how closely the coal quality in a deposit can meet the requirements of future contracts.

QCSMO has the potential to allow the user to change specifications during scheduling. As presently formulated, the user specifies a uniform requirement for all periods throughout the time horizon. If the coal specification changes over time, the ability to change requirements would allow QCSMO to generate a mining sequence that would match the real requirements more closely.

The programs listed in APPENDIX C do not include the function that reads scheduling requirements for each period. QCSMO needs a small program to read the user requirements for each period and put them into the OBJ structure as required for dynamic
scheduling. As mentioned above, QCSMO is a not stand-alone software, so the user has the responsibility to make any changes necessary to employ all the features of QCSMO and to exploit the full potential of the system.

5.2 COMPARING QCSMO WITH ICAMPS

As mentioned in Chapter II, QCSMO interfaces with ICAMPS (Interactive Computer Aided Mine Planning System) for all basic file I/O and calculation operations. ICAMPS is one of the leading commercial softwares in the field of mining operations simulation. Many mines use ICAMPS for long-run and short-range production planning and scheduling. One requirement for the user of ICAMPS is to specify mining sequences for all mining units before starting scheduling. This is usually done by experienced engineers through a cumbersome trial and error process. Once the mining sequences are determined, the user can simulate the mining operations and generate maps and reports.

In order to verify the effectiveness of QCSMO, a comparison between QCSMO and ICAMPS was conducted by the author. The data and grid file for the case study in Chapter IV were used to compare ICAMPS and QCSMO scheduling. In ICAMPS scheduling, all mining sequences are specified by the user before scheduling starts. In QCSMO only the LW mining sequence is specified by the user, but the CM mining sequences are determined during the process of scheduling. In order to adjust coal quality, QCSMO generated the mining sequences for the CM units subject to the user specified LW sequence and the look-ahead constraints while minimizing the deviations from production quantity and quality specifications for each period.

The following table summarizes the comparison of production and quality deviations for ICAMPS and QCSMO. The scheduling period is based on months; twenty working days per month and two working shifts per day are assumed. The required production rates for a LW unit and a CM unit are 100,000 and 20,000 tons per month respectively. Both simulation models are constrained by precedence relations.
<table>
<thead>
<tr>
<th></th>
<th>ICAMPS</th>
<th>QCSMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of periods (month)</td>
<td>79</td>
<td>76</td>
</tr>
<tr>
<td>Working day per month</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Working shifts per day</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Production rate of LW (tons/shift)</td>
<td>2,500</td>
<td>2,500</td>
</tr>
<tr>
<td>Production rate of CM (tons/shift)</td>
<td>500</td>
<td>500</td>
</tr>
<tr>
<td>Total tonnage required per month for LW (ton/month)</td>
<td>100,000</td>
<td>100,000</td>
</tr>
<tr>
<td>Total tonnage mined by LW (tons)</td>
<td>5,074,697</td>
<td>5,153,241</td>
</tr>
<tr>
<td>Average tonnage mined by LW (tons/month)</td>
<td>64,237</td>
<td>67,806</td>
</tr>
<tr>
<td>Standard deviation of tonnage of LW (tons/month)</td>
<td>46,822</td>
<td>44,670</td>
</tr>
<tr>
<td>Total tonnage required for CM (tons/month)</td>
<td>40,000</td>
<td>40,000</td>
</tr>
<tr>
<td>Total tonnage mined by CM units (tons)</td>
<td>1,914,348</td>
<td>1,914,898</td>
</tr>
<tr>
<td>Average tonnage mined by CM (tons/month)</td>
<td>24,232</td>
<td>25,196</td>
</tr>
<tr>
<td>Standard deviation of CM (tons/month)</td>
<td>17,780</td>
<td>15,347</td>
</tr>
<tr>
<td>Quality requirement for sulfur (percent/ton)</td>
<td>3.00</td>
<td>3.00</td>
</tr>
<tr>
<td>Average sulfur content (percent/ton)</td>
<td>3.36</td>
<td>3.34</td>
</tr>
<tr>
<td>Standard deviation of sulfur content (percent/ton)</td>
<td>0.24</td>
<td>0.16</td>
</tr>
<tr>
<td>Quality requirement for ash (percent/ton)</td>
<td>8.00</td>
<td>8.00</td>
</tr>
<tr>
<td>Average ash content (percent/ton)</td>
<td>7.70</td>
<td>7.68</td>
</tr>
<tr>
<td>Standard deviation of ash content (percent/ton)</td>
<td>0.43</td>
<td>0.38</td>
</tr>
</tbody>
</table>

Table 5.1 Comparison between ICAMPS and QCSMO for Whole Mine

Note: Total tons mined by ICAMPS and QCSMO may have some difference because of accumulated round-off and truncation errors.
The results in the above table are quite interesting. One can see in Table 5.1 that the average quality values for both methods are very close because both are scheduling the same mine. However, the standard deviations for the qualities are quite different. For example, the standard deviation of percent sulfur content is about 0.08 less for QCSMO than for ICAMPS. For percent ash content, the standard deviation for QCSMO is 0.05 less. Also QCSMO required three months less than ICAMPS to complete mining. The reason is quite obvious. In ICAMPS scheduling, each mining unit must complete one assigned area before it can start a new one. But in QCSMO scheduling, the CM mining units have more options, which means a CM unit can mine any available area selectively to adjust the quality parameters or shorten the waiting time for a LW units.

<table>
<thead>
<tr>
<th></th>
<th>ICAMPS</th>
<th>QCSMO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of periods (month)</td>
<td>28</td>
<td>28</td>
</tr>
<tr>
<td>Total tonnage mined by LW (tons)</td>
<td>2,574,247</td>
<td>2,638,212</td>
</tr>
<tr>
<td>Average tonnage mined by LW (tons/month)</td>
<td>91,937</td>
<td>94,222</td>
</tr>
<tr>
<td>Standard deviation of tonnage of LW (tons/month)</td>
<td>23,741</td>
<td>9,001</td>
</tr>
<tr>
<td>Total tonnage mined by CM units (tons)</td>
<td>905,245</td>
<td>866,885</td>
</tr>
<tr>
<td>Average tonnage mined by CM (tons/month)</td>
<td>32,330</td>
<td>30,960</td>
</tr>
<tr>
<td>Standard deviation of CM (tons/month)</td>
<td>9,989</td>
<td>9,790</td>
</tr>
<tr>
<td>Quality requirement for sulfur (percent/ton)</td>
<td>3.00</td>
<td>3.00</td>
</tr>
<tr>
<td>Average sulfur content (percent/ton)</td>
<td>3.33</td>
<td>3.32</td>
</tr>
<tr>
<td>Standard deviation of sulfur content (percent/ton)</td>
<td>0.18</td>
<td>0.12</td>
</tr>
<tr>
<td>Quality requirement for ash (percent/ton)</td>
<td>8.00</td>
<td>8.00</td>
</tr>
<tr>
<td>Average ash content (percent/ton)</td>
<td>7.62</td>
<td>7.61</td>
</tr>
<tr>
<td>Standard deviation of ash content (percent/ton)</td>
<td>0.28</td>
<td>0.26</td>
</tr>
</tbody>
</table>

Table 5.2 Comparison between ICAMPS and QCSMO for Steady State
The comparison shown in Table 5.2 is based on the steady state condition. When in steady state, all mining units are working at full capacity. Since the data set is small, the steady state does not last very long, only 28 months. The comparison in Table 5.2 is not quite consistent, because the periods in steady states for ICAMPS and for QCSMO are not always the same although most of them are in common. They correspond to the sections of Figure 5.1 where all mining units are working simultaneously.

The Figure 5.1 is a Gantt chart to show the mining sequences of the LW and CM units for both ICAMPS and QCSMO. The working CM units are denoted as CM1 and CM2. One can see from the chart that QCSMO mines DV areas selectively. This dynamic feature enables the QCSMO CM units to choose the right mining areas in order to adjust the coal quality parameters. The result of such adjustments tends to reduce the standard deviations of the quality as indicated in Table 5.1 and Table 5.2.

One can also see that in Figure 5.1, that QCSMO starts LW mining one month earlier than does ICAMPS. The waiting periods of LW mining unit for QCSMO are also less than ICAMPS. The aggregate effect of QCSMO scheduling is that the time to complete mining is reduced.

The data set used for testing is just for illustration. Though it is not big enough to show the full advantage of QCSMO, the results are still quite impressive. One can expect that if a mine has many CM machines (in this example, only two CM units are used), the quality of coal could be adjusted substantially to meet the requirements and the time a LW unit waits for the next area would be shortened.
Figure 5.1 Gantt Chart showing ICAMPS and QCSMO mining sequences of DV areas mined by CM1 and CM2
5.3 EXPANSION OF QCSMO

Though this research has successfully developed a QCSMO algorithm, there are still some features that need further work. One important improvement, which the author thought possible and necessary, relates to how the deviations are weighted according to their importance and costs. There is an implied assumption that the effect of a deviation varies in a linear manner. For example, in the current model, if the percentage of sulfur
per ton exceeds the requirement by one percent and costs the producer $5 per ton, an additional percent of sulfur will also cost coal producer another $5. In reality, however, that is not the case. If the coal producer has a contract for supplying coal with one percent sulfur content, if the specified coal sulfur content is exceeded by one half percent, it may be acceptable but the price is reduced by $2.5 per ton; As the excess sulfur content increase, the penalty may increase non-linearly. If the sulfur content is two percent more, it may not be accepted at any cost. Obviously, the linear penalty assignment is only true within a very small range of deviations from the requirement.

To make QCSMO more realistic, the objective function needs to be modified to include a non-linear variation of the weights for deviations. To do so, the range of a deviation needs to be broken down into several smaller linear piece-wise intervals in which each deviation range may have a different weight. Also, a corresponding adjustment is needed for each primary constraint that involves a deviation as a decision variable.

The look-ahead constraints also need more work. As discussed in Chapter III, there is a weak point in the MLIP modeling. For the example shown in Figure 5.2, if both the current area LW1 and the next area LW2 are in the scheduling list or ARELIST, then only the current area LW1 has the look-ahead constraints. The mining of the CM units is unconstrained by the DV areas that surround the LW2 because QCSMO only allows one look-ahead constraints for each mining unit. Usually, the impact of lacking look-ahead constraints for one period at the start of a new LW area is negligible. However, if the LW areas are very small, there could be many such transition periods or if the scheduling periods are very long, the time without an active look-ahead constraints could also be long. The impact might be serious.
To solve this problem, the ARELIST and MLIP need to be restructured. To put look-ahead constraints in the MLIP model to monitor the mining of all DV areas during the transition periods is not very difficult, but to handle them in the ARELIST and put it into the LINDO input file could be quite demanding. Also, special features are needed to handle the situation where a LW unit moves to another location or all the LW mining areas are completed.

Figure 5.2 Example where two LW areas are mined during the transition period.
BIBLIOGRAPHY


University of Arizona, Tucson, Arizona, pp A75-A105.


Lonergan, J. E. 1983, Computerized Solutions to Mine Planning and Blending Problems (Coal), Doctoral Dissertation, the University of Arizona.


APPENDIX A

INTER-LANGUAGE CALLS BETWEEN C AND FORTRAN

It is very important to understand the calling conventions of C and FORTRAN before trying to call a FORTRAN routine from C routine or vice versa. A detailed discussion is beyond the scope of this research; those who are interested may refer to any C or FORTRAN book for details. In this appendix, the author lists the major steps for this procedure and an example from QCSMO is provided.

The following fundamentals are required.

1. A uniform programming platform is required for both C and FORTRAN. The author has chosen WATCOM's C and FORTRAN77 compiler and linker as the working platform.

2. Define the function type or subroutine such that they are acceptable by both C and FORTRAN. Make any changes that are required. The calling interface must be simple and without any entangled relation. Passing values must be through parameters or function return, no global variables can be passed directly between C and FORTRAN.

3. File handles defined in one language cannot be passed to the other.

4. A subroutine in FORTRAN has a void type in C. A character type function is not allowed.

5. A proper PRAGMA compiling directive must be used when one routine is going to be called by the other. Interested reader may refer to WATCOM's C&FORTRAN user's manuals for detailed information.

6. Both the calling and called routines need to be recompiled into OBJ file before linking them together.

The following example illustrates the major steps used by QCSMO to call LINDO subroutines written in FORTRAN.
The LINDO subroutines listed below are called by QCSMO to solve MLIP models.

C I HAVE FOLLOWING LINDO SUBROUTINES TO BE CALLED BY QCSMO
C NOTE: THESE SUBROUTINES WERE RECOMPILED BY WATCOM F77 COMPILER
C ILINDO: INVOKE LINDO AND INITIALIZE ITS ENVIRONMENT VARIABLES
  SUBROUTINE ILINDO
C INIT: INITIALIZE LINDO
  SUBROUTINE INIT
C QUIET: SET LINDO DISPLAY MODE. NOISE:-1 NO DISPLAY;
  0 TERSE; 1 VERBOSE; 2 MASSAGE AT EACH PIVOT STEP
  SUBROUTINE QUIET( NOISE )
C LINIO: READ MPS FILE LININ, SOLVE MLIP MODEL, WRITE LNDOUT FILE
  SUBROUTINE LINIO

/* In C I need to do ---*/
#pragma aux ilindo nAn; /* compiling directives */
#pragma aux init nAn; /* that changes the calling */
#pragma aux quiet nAn; /* convention of FORTRAN to */
#pragma aux linio nAn; /* C convention */
/* define FORTRAN subroutine as C void function */
/* INTEGER -- long; REAL -- double; */
void ilindo(void);
void init(void);
void quiet(long *);
void linio(void);
/* In main function I can call those FORTRAN routines as if
they were written in C. Note, these C routines were compiled by
WATCOM C compiler */
main()
{......
ilindo();
init();
quiet(0);
linio();
......
}

The following FORTRAN routine LNDIO.FOR was written for LINDO INPUT
AND OUTPUT. Note that the file handles in C and FORTRAN are different, so special
efforts are needed to manage the files.

C GO IS A LINDO SUBROUTINE THAT SOLVES THE MLIP MODEL
  SUBROUTINE LNDIO
  INTEGER ISTATE
  CALL RDLND
  CALL GO(0,ISTATE)
  CALL SVLND
  RETURN
  END
SUBROUTINE RDLND
OPEN (UNIT = 40, FILE = 'LNDIN', STATUS = 'OLD', ERR = 10)
C RDMPS IS A LINDO SUBROUTINE THAT READ IN MPS FORMATTED MODEL
CALL RDMPS(40,1)
CLOSE(40)
RETURN
10 PRINT *, 'Error in opening an input file'
RETURN
END

SUBROUTINE SVLND
OPEN (UNIT = 41, FILE = 'LNDOUT', STATUS = 'OLD', ERR=20)
CLOSE (41,STATUS='DELETE')
20 OPEN (UNIT = 41, FILE = 'LNDOUT', STATUS='NEW')
C SDBC IS A LINDO SUBROUTINE THAT SAVE A SOLUTION IN DBC FORMAT
CALL SDBC(41)
CLOSE (41)
RETURN
END

Note: Other examples are given in the WATCOM F77 user's manuals.
APPENDIX B

QCSMO INPUT AND OUTPUT FILES

OBJECT.DEF

000013301003020600302
50000 20000 50000 10000 3.0 8 0 0
1 2 1 0.5 2 2 0.5 0.5 3 25 5 25

RELATION

000013300001430100302002010044202004420100452020045200004620000472
00001430000153000046203003020004720100482020048200004920000502
00001530000000040030200004920100512020051200005020000522
010030202003020100442
02003020300302000462
01004420200442
02004420100452
000046202004520000472
00004720100482
030030204003020000492
000049202004820000502
00005020100512
04003020000522
00005220200512
0100452000000
0200452000000
0100482000000
0200482000000
0100512000000
0200512000000
06003020000582
00005820000592
0000592000000

GC (CONVERTED FROM WORK.GC BY LOOK.EXE)

000100 start S 50000 20000 3.00 8.00 0.00 0.00
000101 0100302 C 10000 10000 3.78 8.23 7796.09 2337.84
000102 0600302 C 10000 10000 4.39 8.93 13858.79 9274.65
000103 ending E 0 20000 1.08 0.58 21654.88 11612.49
000200 start S 50000 20000 3.00 8.00 0.00 0.00
000201 0100302 C 10000 808 3.74 8.28 596.41 227.20
000202 0600302 C 10000 808 4.43 8.94 1152.96 757.77
000203 0200302 C 10000 9192 3.73 8.29 6710.99 2646.27
000204 0100442 C 10000 9192 3.64 8.03 5910.39 293.20
000205 ending E 0 20000 0.72 0.10 14370.76 3924.45
000300 start S 50000 20000 3.00 8.00 0.00 0.00
000301 0200302 C 10000 10000 3.69 8.33 6852.48 3277.80
000302 0100442 C 10000 10000 3.54 7.88 5396.75 618.55
000303 ending E 0 20000 0.61 0.10 12249.23 2040.71
000400 start S 50000 20000 3.00 8.00 0.00 0.00
000401 0200302 C 10000 10000 3.64 8.37 6399.84 3659.60
000402 0100442 C 10000 3521 3.47 7.77 1664.66 407.68
| 000403 | 0200442 C | 10000 | 6479 | 3.43 | 7.69 | 2783.46 | 994.82 |
| 000404 | ending E | 0 | 20000 | 0.54 | 0.04 | 10847.97 | 854.59 |
| 000500 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 000501 | 0200302 C | 10000 | 10000 | 3.61 | 8.39 | 6053.21 | 3940.69 |
| 000502 | 0200442 C | 10000 | 10000 | 3.37 | 7.58 | 3692.39 | 2104.23 |
| 000503 | ending E | 0 | 20000 | 0.49 | -0.01 | 9745.60 | 133.88 |
| 000600 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 000601 | 0200302 C | 10000 | 10000 | 3.61 | 8.39 | 6053.21 | 3940.69 |
| 000602 | 0200442 C | 10000 | 10000 | 3.37 | 7.58 | 3692.39 | 2104.23 |
| 000603 | ending E | 0 | 20000 | 0.49 | -0.01 | 9745.60 | 133.88 |
| 000700 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 000701 | 0200302 C | 10000 | 10000 | 3.58 | 8.42 | 5770.71 | 4215.01 |
| 000702 | 0200442 C | 10000 | 10000 | 3.31 | 7.45 | 3109.48 | 2740.77 |
| 000703 | ending E | 0 | 20000 | 0.44 | -0.06 | 8880.19 | 633.27 |
| 000800 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 000801 | 0200442 C | 10000 | 10000 | 3.25 | 7.26 | 2530.16 | 3711.61 |
| 000802 | 0000462 C | 10000 | 10000 | 3.38 | 7.99 | 3821.96 | 35.06 |
| 000803 | ending E | 0 | 20000 | 0.32 | -0.37 | 6352.12 | 3746.66 |
| 000900 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 000901 | 0200442 C | 10000 | 10000 | 3.25 | 7.19 | 2492.51 | 4042.80 |
| 000902 | 0000462 C | 10000 | 10000 | 3.29 | 7.75 | 2734.58 | 3267.37 |
| 000903 | ending E | 0 | 20000 | 0.27 | -0.52 | 5439.95 | 5204.69 |
| 001000 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 001001 | 0200442 C | 10000 | 10000 | 3.26 | 7.14 | 2581.60 | 4300.24 |
| 001002 | 0000462 C | 10000 | 10000 | 3.23 | 7.75 | 2322.13 | 1161.88 |
| 001003 | ending E | 0 | 20000 | 0.25 | -0.65 | 4903.73 | 6458.94 |
| 001100 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 001101 | 0200442 C | 10000 | 10000 | 3.28 | 7.11 | 2777.39 | 4473.07 |
| 001102 | 0000462 C | 10000 | 10000 | 3.20 | 7.99 | 3821.96 | 35.06 |
| 001103 | ending E | 0 | 20000 | 0.32 | -0.37 | 6352.12 | 3746.66 |
| 001200 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 001201 | 0200442 C | 10000 | 10000 | 3.30 | 7.08 | 3049.98 | 4591.12 |
| 001202 | 0000462 C | 10000 | 10000 | 3.19 | 7.28 | 1896.87 | 3614.95 |
| 001203 | ending E | 0 | 20000 | 0.25 | -0.82 | 4946.85 | 8206.07 |
| 001300 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 001301 | 0200442 C | 10000 | 10000 | 3.28 | 7.11 | 2777.39 | 4473.07 |
| 001302 | 0200452 C | 10000 | 10000 | 3.20 | 7.18 | 2002.02 | 4113.89 |
| 001303 | 0100452 C | 10000 | 10000 | 3.23 | 7.57 | 2322.13 | 2158.70 |
| 001304 | ending E | 0 | 20000 | 0.25 | -0.65 | 4903.73 | 6458.94 |
| 001400 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 001401 | 0300302 C | 10000 | 10000 | 3.20 | 7.41 | 1982.42 | 2968.13 |
| 001402 | 0000462 C | 10000 | 10000 | 3.20 | 7.99 | 3821.96 | 35.06 |
| 001403 | ending E | 0 | 20000 | 0.24 | -0.74 | 4759.81 | 7441.20 |
| 001500 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 001501 | 0300302 C | 10000 | 10000 | 3.20 | 7.41 | 1982.42 | 2968.13 |
| 001502 | 0000462 C | 10000 | 10000 | 3.20 | 7.99 | 3821.96 | 35.06 |
| 001503 | ending E | 0 | 20000 | 0.24 | -0.74 | 4759.81 | 7441.20 |
| 001600 | start S | 50000 | 20000 | 3.00 | 8.00 | 0.00 | 0.00 |
| 001601 | 0300302 C | 10000 | 10000 | 3.20 | 7.41 | 1982.42 | 2968.13 |
| 001602 | 0000462 C | 10000 | 10000 | 3.20 | 7.99 | 3821.96 | 35.06 |
| 001603 | ending E | 0 | 20000 | 0.24 | -0.74 | 4759.81 | 7441.20 |
001702 000472 C 10000 10000 3.32 6.98 3152.36 5107.83
001703 ending E 0 20000 0.43 -0.27 8515.09 2717.38
001800 start S 50000 20000 3.00 8.00 0.00 0.00
001801 0199993 L 50000 10 3.00 8.00 0.00 0.00
001802 0000133 L 50000 49990 3.25 7.07 12711.80 23128.61
001803 0000472 C 10000 10000 3.33 6.96 1796.11 2786.65
001804 ending E 50000 20000 0.30 -0.74 21266.24 25815.99
001805 0100482 C 10000 4636 3.32 6.95 1479.29 2425.76
001806 0000472 C 10000 5364 3.33 6.96 1796.11 2786.65
001807 0000522 C 10000 9343 3.67 8.49 6278.27 4561.21
001808 ending E 0 20000 0.47 -0.10 9368.65 979.40
001900 start S 50000 20000 3.00 8.00 0.00 0.00
001901 0600302 C 10000 10000 3.53 8.53 5267.70 5323.05
001902 0100482 C 10000 7211 3.32 6.95 2276.09 3785.24
001903 ending E 0 20000 0.59 0.02 11746.06 422.28
002000 start S 50000 20000 3.00 8.00 0.00 0.00
002001 0600302 C 10000 10000 4.53 8.96 15291.08 9626.08
002002 0300302 C 10000 10000 3.53 8.56 5330.22 5598.90
002003 ending E 0 20000 1.03 0.76 20621.30 15224.97
002100 start S 50000 20000 3.00 8.00 0.00 0.00
002101 0600302 C 10000 10000 3.63 8.99 6383.16 3863.17
002102 0300302 C 10000 10000 3.54 8.58 2127.19 2282.15
002103 ending E 0 20000 0.73 0.59 14691.22 11863.59
002200 start S 50000 20000 3.00 8.00 0.00 0.00
002201 0400302 C 10000 10000 3.67 8.62 5692.69 6153.91
002202 0000492 C 10000 10000 3.41 8.16 4066.58 1592.44
002203 ending E 0 20000 0.49 0.39 9759.27 7746.35
002300 start S 50000 20000 3.00 8.00 0.00 0.00
002301 0400302 C 10000 10000 3.70 8.70 6054.39 6470.99
002302 0000492 C 10000 10000 3.26 7.94 2579.04 2679.04
002303 ending E 0 20000 0.47 0.29 9489.60 5856.67
002400 start S 50000 20000 3.00 8.00 0.00 0.00
002401 0400302 C 10000 10000 3.76 8.73 6479.59 6749.61
002402 0000492 C 10000 10000 3.26 7.74 2557.81 2893.37
002403 ending E 0 20000 0.51 0.08 10120.71 1520.03
002500 start S 50000 20000 3.00 8.00 0.00 0.00
002501 0400302 C 10000 10000 3.76 8.70 6979.31 7030.73
002502 0000492 C 10000 10000 3.26 7.57 2679.18 2157.02
002503 ending E 0 20000 0.48 0.14 9658.49 2716.68
002600 start S 50000 20000 3.00 8.00 0.00 0.00
002601 0400302 C 10000 10000 3.76 8.73 7562.90 7306.76
002602 0000492 C 10000 10000 3.26 7.42 2557.81 2893.37
002603 ending E 0 20000 0.51 0.08 10120.71 1520.03
002700 start S 50000 20000 3.00 8.00 0.00 0.00
002701 0400302 C 10000 10000 3.78 8.74 513.73 487.71
002702 0000492 C 10000 10000 3.26 7.30 2576.64 3503.86
002703 0000522 C 10000 9343 3.67 8.49 6278.27 4561.21
002704 ending E 0 20000 0.47 -0.10 9368.65 979.40
002800 start S 50000 20000 3.00 8.00 0.00 0.00
002801 0000492 C 10000 10000 3.27 7.20 2681.65 4023.47
002802 0000522 C 10000 10000 3.61 8.27 6504.39 6470.99
002803 ending E 0 20000 0.47 -0.10 9368.65 979.40
002900 start S 50000 20000 3.00 8.00 0.00 0.00
002901 0000492 C 10000 10000 3.28 7.11 2843.35 4468.52
002902 0000522 C 10000 10000 3.55 8.07 5493.01 692.20
002903 ending E 0 20000 0.42 -0.41 8336.36 4122.42
003000 start S 50000 20000 3.00 8.00 0.00 0.00
<table>
<thead>
<tr>
<th>Start Index</th>
<th>End Index</th>
<th>Start Time</th>
<th>End Time</th>
<th>Start Value</th>
<th>End Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>003001</td>
<td>003001</td>
<td>0000133</td>
<td>0000133</td>
<td>50000</td>
<td>27811</td>
</tr>
<tr>
<td>003002</td>
<td>003002</td>
<td>0000492</td>
<td>0000492</td>
<td>10000</td>
<td>8328</td>
</tr>
<tr>
<td>003003</td>
<td>003003</td>
<td>0000522</td>
<td>0000522</td>
<td>10000</td>
<td>8328</td>
</tr>
<tr>
<td>003004</td>
<td>003004</td>
<td>ending</td>
<td>E</td>
<td>27811</td>
<td>20000</td>
</tr>
<tr>
<td>003100</td>
<td>003100</td>
<td>start</td>
<td>S</td>
<td>50000</td>
<td>20000</td>
</tr>
<tr>
<td>003101</td>
<td>003101</td>
<td>0000133</td>
<td>0000133</td>
<td>50000</td>
<td>46688</td>
</tr>
<tr>
<td>003102</td>
<td>003102</td>
<td>0000492</td>
<td>0000492</td>
<td>10000</td>
<td>8328</td>
</tr>
<tr>
<td>003103</td>
<td>003103</td>
<td>0000522</td>
<td>0000522</td>
<td>10000</td>
<td>8328</td>
</tr>
<tr>
<td>003104</td>
<td>003104</td>
<td>0200482</td>
<td>0200482</td>
<td>10000</td>
<td>1672</td>
</tr>
<tr>
<td>003105</td>
<td>003105</td>
<td>0000502</td>
<td>0000502</td>
<td>10000</td>
<td>1672</td>
</tr>
<tr>
<td>003106</td>
<td>003106</td>
<td>ending</td>
<td>E</td>
<td>46688</td>
<td>20000</td>
</tr>
<tr>
<td>003200</td>
<td>003200</td>
<td>start</td>
<td>S</td>
<td>50000</td>
<td>20000</td>
</tr>
<tr>
<td>003201</td>
<td>003201</td>
<td>0000133</td>
<td>0000133</td>
<td>50000</td>
<td>50000</td>
</tr>
<tr>
<td>003202</td>
<td>003202</td>
<td>0000492</td>
<td>0000492</td>
<td>10000</td>
<td>8328</td>
</tr>
<tr>
<td>003203</td>
<td>003203</td>
<td>0000522</td>
<td>0000522</td>
<td>10000</td>
<td>8328</td>
</tr>
<tr>
<td>003204</td>
<td>003204</td>
<td>0000502</td>
<td>0000502</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003205</td>
<td>003205</td>
<td>ending</td>
<td>E</td>
<td>50000</td>
<td>20000</td>
</tr>
<tr>
<td>003300</td>
<td>003300</td>
<td>start</td>
<td>S</td>
<td>50000</td>
<td>20000</td>
</tr>
<tr>
<td>003301</td>
<td>003301</td>
<td>0000133</td>
<td>0000133</td>
<td>50000</td>
<td>40701</td>
</tr>
<tr>
<td>003302</td>
<td>003302</td>
<td>0000492</td>
<td>0000492</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003303</td>
<td>003303</td>
<td>0000522</td>
<td>0000522</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003304</td>
<td>003304</td>
<td>ending</td>
<td>E</td>
<td>40701</td>
<td>20000</td>
</tr>
<tr>
<td>003400</td>
<td>003400</td>
<td>start</td>
<td>S</td>
<td>50000</td>
<td>20000</td>
</tr>
<tr>
<td>003401</td>
<td>003401</td>
<td>0000133</td>
<td>0000133</td>
<td>50000</td>
<td>40701</td>
</tr>
<tr>
<td>003402</td>
<td>003402</td>
<td>0000492</td>
<td>0000492</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003403</td>
<td>003403</td>
<td>0000522</td>
<td>0000522</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003404</td>
<td>003404</td>
<td>ending</td>
<td>E</td>
<td>40701</td>
<td>20000</td>
</tr>
<tr>
<td>003500</td>
<td>003500</td>
<td>start</td>
<td>S</td>
<td>50000</td>
<td>20000</td>
</tr>
<tr>
<td>003501</td>
<td>003501</td>
<td>0000133</td>
<td>0000133</td>
<td>50000</td>
<td>40701</td>
</tr>
<tr>
<td>003502</td>
<td>003502</td>
<td>0000492</td>
<td>0000492</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003503</td>
<td>003503</td>
<td>0000522</td>
<td>0000522</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003504</td>
<td>003504</td>
<td>ending</td>
<td>E</td>
<td>40701</td>
<td>20000</td>
</tr>
<tr>
<td>003600</td>
<td>003600</td>
<td>start</td>
<td>S</td>
<td>50000</td>
<td>20000</td>
</tr>
<tr>
<td>003601</td>
<td>003601</td>
<td>0000133</td>
<td>0000133</td>
<td>50000</td>
<td>40701</td>
</tr>
<tr>
<td>003602</td>
<td>003602</td>
<td>0000492</td>
<td>0000492</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003603</td>
<td>003603</td>
<td>0000522</td>
<td>0000522</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003604</td>
<td>003604</td>
<td>ending</td>
<td>E</td>
<td>40701</td>
<td>20000</td>
</tr>
<tr>
<td>003700</td>
<td>003700</td>
<td>start</td>
<td>S</td>
<td>50000</td>
<td>20000</td>
</tr>
<tr>
<td>003701</td>
<td>003701</td>
<td>0000133</td>
<td>0000133</td>
<td>50000</td>
<td>40701</td>
</tr>
<tr>
<td>003702</td>
<td>003702</td>
<td>0000522</td>
<td>0000522</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003703</td>
<td>003703</td>
<td>0000502</td>
<td>0000502</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003704</td>
<td>003704</td>
<td>ending</td>
<td>E</td>
<td>40701</td>
<td>20000</td>
</tr>
<tr>
<td>003800</td>
<td>003800</td>
<td>start</td>
<td>S</td>
<td>50000</td>
<td>20000</td>
</tr>
<tr>
<td>003801</td>
<td>003801</td>
<td>0000133</td>
<td>0000133</td>
<td>50000</td>
<td>40701</td>
</tr>
<tr>
<td>003802</td>
<td>003802</td>
<td>0000522</td>
<td>0000522</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003803</td>
<td>003803</td>
<td>0000502</td>
<td>0000502</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003804</td>
<td>003804</td>
<td>ending</td>
<td>E</td>
<td>40701</td>
<td>20000</td>
</tr>
<tr>
<td>003900</td>
<td>003900</td>
<td>start</td>
<td>S</td>
<td>50000</td>
<td>20000</td>
</tr>
<tr>
<td>003901</td>
<td>003901</td>
<td>0000133</td>
<td>0000133</td>
<td>50000</td>
<td>40701</td>
</tr>
<tr>
<td>003902</td>
<td>003902</td>
<td>0000522</td>
<td>0000522</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003903</td>
<td>003903</td>
<td>0000502</td>
<td>0000502</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>003904</td>
<td>003904</td>
<td>ending</td>
<td>E</td>
<td>40701</td>
<td>20000</td>
</tr>
<tr>
<td>004000</td>
<td>004000</td>
<td>start</td>
<td>S</td>
<td>50000</td>
<td>20000</td>
</tr>
<tr>
<td>004001</td>
<td>004001</td>
<td>0000133</td>
<td>0000133</td>
<td>50000</td>
<td>40701</td>
</tr>
<tr>
<td>004002</td>
<td>004002</td>
<td>0000522</td>
<td>0000522</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>004003</td>
<td>004003</td>
<td>0000502</td>
<td>0000502</td>
<td>10000</td>
<td>2500</td>
</tr>
<tr>
<td>004004</td>
<td>004004</td>
<td>ending</td>
<td>E</td>
<td>40701</td>
<td>20000</td>
</tr>
<tr>
<td>ID</td>
<td>Symbol</td>
<td>Value 1</td>
<td>Value 2</td>
<td>Value 3</td>
<td>Value 4</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>004100</td>
<td>start S</td>
<td>50000</td>
<td>20000</td>
<td>3.00</td>
<td>8.00</td>
</tr>
<tr>
<td>004101</td>
<td>0000133</td>
<td>L</td>
<td>50000</td>
<td>40000</td>
<td>3.35</td>
</tr>
<tr>
<td>004102</td>
<td>0000522</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
</tr>
<tr>
<td>004103</td>
<td>0000502</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
</tr>
<tr>
<td>004104</td>
<td>ending E</td>
<td>40000</td>
<td>20000</td>
<td>0.23</td>
<td>-0.17</td>
</tr>
<tr>
<td>004200</td>
<td>start S</td>
<td>50000</td>
<td>20000</td>
<td>3.00</td>
<td>8.00</td>
</tr>
<tr>
<td>004201</td>
<td>0000133</td>
<td>L</td>
<td>50000</td>
<td>40000</td>
<td>3.39</td>
</tr>
<tr>
<td>004202</td>
<td>0000522</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
</tr>
<tr>
<td>004203</td>
<td>0000502</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
</tr>
<tr>
<td>004204</td>
<td>ending E</td>
<td>40000</td>
<td>20000</td>
<td>0.26</td>
<td>-0.11</td>
</tr>
<tr>
<td>004300</td>
<td>start S</td>
<td>50000</td>
<td>20000</td>
<td>3.00</td>
<td>8.00</td>
</tr>
<tr>
<td>004301</td>
<td>0000133</td>
<td>L</td>
<td>50000</td>
<td>40000</td>
<td>3.44</td>
</tr>
<tr>
<td>004302</td>
<td>0000522</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
</tr>
<tr>
<td>004303</td>
<td>0000502</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
</tr>
<tr>
<td>004304</td>
<td>ending E</td>
<td>40000</td>
<td>20000</td>
<td>0.29</td>
<td>-0.04</td>
</tr>
<tr>
<td>004400</td>
<td>start S</td>
<td>50000</td>
<td>20000</td>
<td>3.00</td>
<td>8.00</td>
</tr>
<tr>
<td>004401</td>
<td>0000133</td>
<td>L</td>
<td>50000</td>
<td>33576</td>
<td>3.49</td>
</tr>
<tr>
<td>004402</td>
<td>0000522</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
</tr>
<tr>
<td>004403</td>
<td>0000502</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
</tr>
<tr>
<td>004404</td>
<td>ending E</td>
<td>33576</td>
<td>20000</td>
<td>0.31</td>
<td>0.02</td>
</tr>
<tr>
<td>004500</td>
<td>start S</td>
<td>50000</td>
<td>20000</td>
<td>3.00</td>
<td>8.00</td>
</tr>
<tr>
<td>004501</td>
<td>0000133</td>
<td>L</td>
<td>50000</td>
<td>14481</td>
<td>3.51</td>
</tr>
<tr>
<td>004502</td>
<td>0000522</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
</tr>
<tr>
<td>004503</td>
<td>0000143</td>
<td>L</td>
<td>50000</td>
<td>35519</td>
<td>3.30</td>
</tr>
<tr>
<td>004504</td>
<td>ending E</td>
<td>50000</td>
<td>20000</td>
<td>0.40</td>
<td>-0.54</td>
</tr>
<tr>
<td>004600</td>
<td>start S</td>
<td>50000</td>
<td>20000</td>
<td>3.00</td>
<td>8.00</td>
</tr>
<tr>
<td>004601</td>
<td>0600302</td>
<td>C</td>
<td>10000</td>
<td>8340</td>
<td>4.67</td>
</tr>
<tr>
<td>004602</td>
<td>0000522</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>3.00</td>
</tr>
<tr>
<td>004603</td>
<td>0000143</td>
<td>L</td>
<td>50000</td>
<td>50000</td>
<td>3.29</td>
</tr>
<tr>
<td>004604</td>
<td>ending E</td>
<td>50000</td>
<td>20000</td>
<td>0.40</td>
<td>-0.54</td>
</tr>
<tr>
<td>004700</td>
<td>start S</td>
<td>50000</td>
<td>20000</td>
<td>3.00</td>
<td>8.00</td>
</tr>
<tr>
<td>004701</td>
<td>0600302</td>
<td>C</td>
<td>10000</td>
<td>10000</td>
<td>4.78</td>
</tr>
<tr>
<td>004702</td>
<td>0000522</td>
<td>C</td>
<td>10000</td>
<td>1203</td>
<td>3.00</td>
</tr>
<tr>
<td>004703</td>
<td>0000143</td>
<td>L</td>
<td>50000</td>
<td>50000</td>
<td>3.29</td>
</tr>
<tr>
<td>004704</td>
<td>ending E</td>
<td>50000</td>
<td>14468</td>
<td>0.49</td>
<td>-0.54</td>
</tr>
<tr>
<td>004800</td>
<td>start S</td>
<td>50000</td>
<td>20000</td>
<td>3.00</td>
<td>8.00</td>
</tr>
<tr>
<td>004801</td>
<td>0600302</td>
<td>C</td>
<td>10000</td>
<td>5395</td>
<td>5.01</td>
</tr>
<tr>
<td>004802</td>
<td>0000143</td>
<td>L</td>
<td>50000</td>
<td>50000</td>
<td>3.27</td>
</tr>
<tr>
<td>004803</td>
<td>ending E</td>
<td>50000</td>
<td>20000</td>
<td>0.55</td>
<td>-0.54</td>
</tr>
<tr>
<td>004900</td>
<td>start S</td>
<td>50000</td>
<td>20000</td>
<td>3.00</td>
<td>8.00</td>
</tr>
<tr>
<td>004901</td>
<td>0600302</td>
<td>C</td>
<td>10000</td>
<td>5395</td>
<td>5.01</td>
</tr>
<tr>
<td>004902</td>
<td>0000143</td>
<td>L</td>
<td>50000</td>
<td>50000</td>
<td>3.27</td>
</tr>
<tr>
<td>004903</td>
<td>ending E</td>
<td>50000</td>
<td>20000</td>
<td>0.55</td>
<td>-0.54</td>
</tr>
</tbody>
</table>
NOTE: To use LOOK.EXE, just type `LOOK GC` at DOS prompt. The BTree file WORK.GC will be converted to ASCII file GC. Look at the QPLAN.H file for more information contained in WORK.GC. To convert all the contents of WORK.GC, proper modification is needed on LOOK.C.
APPENDIX C
PROGRAM LISTING

The following programs were specially written for QCSMO.

QPLAN.H
QPMAIN.C
QPLAN.C
INI_QP.C
WT_LINDO.C
RD_LINDO.C
WT_GC.C
GRAF.C
LOOK.C

QPLAN.H

/* QPLAN.H is the head file to support the QCSMO underground mining
scheduling algorithm
Yang Bai 1994, Finalized
ISE dept. Ohio Univ. */

/* global variables definition */
#ifndef MXU
#define MXU 20
#endif
#define EPSILON 0.1 /* accuracy factor for iteration of searching shifts */
#define MAXITER 50 /* Maximum number of iteration allowed */
#define MXDV 10 /* Maximum number of development areas around LW */
#define MINTON 1 /* Minimum tonnage that is negligible */
#define SCALE 100 /* Minimum counting unit */
#define BIG_M 9999999 /* Infinity used to select mutually exclusive cons. */
#define PENALTY 9999999 /* penalty coefficient in objective for LW slk var */

/* global files' handle definition */
extern int fpqp;
extern int fpgc;

/* definition of file structure */

/* 1. define a Quality Planning file (qp file)
This is a temporary file used in scheduling */

struct qptype {
    char areseq[8];    /* Area Identification */
    char aretype;     /* Area type L or C */
    char arenxt[8];   /* Area followed only for LW */
    char status;      /* Mining status U: undone A: available
                        M: mining or D: done*/
    double mpx[4];   /* X-coordinates of 4 corners */
    double mpy[4];   /* Y-coordinates of 4 corners */
    char adjare[MXDV][8]; /* adjacent DV areas No. 1,2,3,4... for LW
                        or immediately followed areas for DV */
    double totton;   /* Total tons of coal reserve of an area */
double tadv; /* Total length of an area on mid-line */
double qpara[4]; /* Average quality parameters of an area */
double pext; /* from SE file percentage of extraction */
};

/* 2. define a Gantt Chart output file ( gc file )
the field unitseq and areseq may leave blank when the whole row
shows the general information about a period */

struct gctype {
    char pd[7]; /* period No. Max period ->9999|99< Max areas in
                 this period */
    char areseq[8]; /* area identification No.
                     or marked starting or ending of a period */
    char aretype; /* area type */
    double rtons; /* production tonnage requirement */
    double mtons; /* actual mined tons */
    double qpara[4]; /* production quality standard requirements
                    or actual quality of an area */
    double cost[4]; /* deviation cost */
};

/* 3. data struct representing production requirements each period */
typedef struct {
    char st_are[MXU][8]; /* starting areas */
    double ttons_lw; /* total tonnage required for all LW units */
    double ttons_cm; /* total tonnage required for all CM units */
    double tons_lw; /* tonnage objective for each LW unit */
    double tons_cm; /* tonnage objective for each CM unit */
    double p_wt[6];
    double n_wt[6]; /* cost weights for positive & negative deviations.
    double qpara[4]; /* quality requirement parameters
                     if 0 means don't care */
    double q_dev[4]; /* quality deviations */
} OBJECT; /* objective */

/* 4. data struct representing development areas */
typedef struct {
    char seq[8]; /* development area surrounding next LW area */
    char status; /* mine status of the area */
    double ttons; /* total tonnage reserve of an area */
} DV;

/* 5. data struct representing currently mined area for both LW and DV */
typedef struct are_tag {
    char seq[8]; /* area ID of a currently mined area */
    char type; /* mining unit type 'L': LW 'C': CM */
    char status; /* area status: available, M: mining, A: available, D: will be Done */
    char nxt[8]; /* next area for long wall */
    char lndnm[9]; /* lindo variable name for this area */
    double px[4]; /* X-Y coordinates of an area */
    double py[4]; /* according to advance in a period */
    double mpx[4]; /* whole area coordinates */
    double mpy[4]; /* whole area coordinates */
    double rtons; /* required tonnage per period by the user */
    double mtons; /* mined tons in current period. If an area is 'M',
                   it = rtons, if 'D', it = remaining tons in an area */
    double ttons; /* total tonnage reserve of an area */
    double ptons; /* previous period ending mined tons */
double ctons;       /* current period ending mined tons */
double tadv;        /* total length of an area */
double cadv;        /* current period accumulated advance */
double padv;        /* previous period ending advance */
double pext;        /* percentage of extraction from SE file */
double qpara[4];    /* average quality parameters of an area according to
tonnage assigned */
DV dv[MXDV];        /* DV areas surrounding a LW or possible succeeding
areas of a DV */
struct are_tag      *next;  /* next area pointer */
)
AREA;

defined struct list_tag
{
    AREA *head;      /* pointer to the head of area list */
    AREA *tail;      /* pointer to the last element of the list */
} ARELIST;

/* external functions from ICAMPS LIB or OBJ */
extern void dadv_line(long iteration, double mpx[], double mpy[],
    double padv, double cadv, double tadv,
    double xout[], double yout[]);
extern long prop_area_grid(double xb[], double yb[], long isect, long lnum,
    long lgrid, long idiff, long imach);
extern void chk_para(AREA *area, long *flg);
extern void calc_tons(AREA *area);
extern void calc_tons(AREA *area);
extern void error(char msg[]);
extern int ctol(char cc);
extern char *btGetErr(void);
#include "d:\progll\quick\icampsll.h"
#include "d:\yang\thesis\qplan.h"
int FLAG; /* flag controlling how many periods to go */
int fpqp; /* btree file handle for QP, GC file */
int fpqc;
void qplan(void);

QPMAIN.C

#include "d:\progll\quick\icampsll.h"
#include "d:\yang\thesis\qplan.h"
int FLAG; /* flag controlling how many periods to go */
int fpqp; /* btree file handle for QP, GC file */
int fpqc;
void qplan(void);

/*---------------------------------------------------------------
Main function:
In the main function, the environment variables are set to proper
initial condition.
Then, the qplan() function does all the job for QCSMO.
---------------------------------------------------------------*/

void main(int argc, char* argv[])
{
    long i;
    char filepath[80], filepath2[80];
    _notekmode = 7447;
    DEBUG = 0;
    FLAG = atoi(argv[1]); /* get user given period number */
    /* read the default names of the grid and calendar files ......*/
    if ((fpfd = fopen("FILES.DEF", "r")) != NULL)
{ 
    fscanf(fpfd, "%s", CALEND_NAME);
    for (i =0; i < 20; i++) fscanf(fpfd, "%s", GRID_NAME[i]);
    fscanf(fpfd, "%s", FNAMCUR);
    fclose(fpfd);
} 

else 
{
    strcpy(CALEND_NAME, "CALEND.CL");
    for (i=0; i < 20; i++) strcpy(GRID_NAME[i], "Not_yet_defined");
}

strcpy(filepath,FNAMCUR);
strcat(filepath, "QUALITY.DEF");
if ((FPQD = fopen(filepath, "r")) != NULL)
{
    fscanf(FPQD, "%lf", &DT1);
    fscanf(FPQD, "%lf", &DD3);
    fscanf(FPQD, "%lf", &DD5);
    fscanf(FPQD, "%lf", &DD2);
    fscanf(FPQD, "%lf", &DD4);
    fscanf(FPQD, "%lf", &DD6);
    fscanf(FPQD, "%lf", &DD7);
    fscanf(FPQD, "%lf", &DT3);
}
qplan();
}

QPLAN.C

#include "d:\prog11\quick\icamps11.h"
#include "d:\yang\thesis\qplan.h"
#include <malloc.h>
#include <graph.h>
#include <conio.h>

/*------------------
LINDO interface FORTRAN functions ----------*/
#pragma aux ilindo "A"
#pragma aux init "All"
#pragma aux quiet "IIAII"
#pragma aux Indio "IIAII"
void ilindo(void);
void init(void);
void quiet( long *);
void Indio(void);

/*------------------ C functions ---------------------*/
static long add_area( ARELIST *arelist, AREA *area);
static AREA *ini_area( char areseq[] );
static void ini_list(ARELIST *arelist, OBJECT obj);
void open_object( OBJECT * obj);
void error( char msg[] );
void open_qp(void);
void graf();
void wt_gc(ARELIST *arelist,OBJECT *obj);
AREA * inlist(ARELIST *arelist, char areseq[] );
extern void chk_status(ARELIST *arelist, char areseq[], char new_status);
extern FLAG;
extern char *1_a(long);

Function qplan() Coded by Yang Bai, 12,1993

Program that schedules all mining units and optimally assign areas
to each unit so that in each period the production quantity and quality requirements will be met and at the end of scheduling the whole mine will be efficiently mined.

files being opened in this routine
area definition file : (WORK.SE)
temporary working file : (WORK.GP)
output file : (WORK.GC)
-----------------------------------------------------------------------*/

doctrine { plan()
    /* ------------------ VCARIABLES DEFINITION ------------------ */

    /* system structure definition */
    ARELIST *arelist, list;
    AREA area, *areptr,*aretmp;
    OBJECT obj;

    /* structure definition */
    struct setype sef;
    struct qytype qyf;
    struct qptype qpf;
    struct gctype gcf;

    /* file handles */
    FILE *rel_ptr, *fp_obj;
    char buffer[100];
    char tmpare[5][8];

    /* system variables */
    char filepath[80];
    long lw_count = 0;
    long i, j, fIg, l_tmp;

    /* ---- STARTING PROGRAM ---- */

    /* initializing scheduling environment */
    arelist = &list;
    arelist->head = NULL;
    strcpy(FNAMECUR," ");

    /* open and read in object.def, user defined objective file */
    open_object(&obj);

    /* create and initialize file qp (quality plan )*/
    open_qp();

    /* show mining on screen */
    graf();

    /* creating output file gc (Gannt-chart output). */
    strcpy(filepath,FNAMECUR);
    strcat(filepath,"WORK.GCD");
    remove(filepath);
    strcpy(filepath,FNAMECUR);
    strcat(filepath,"WORK.GC");
    remove(filepath);

    strcpy(filepath,FNAMECUR);
    strcat(filepath,"WORK.GC");
    if(bt_create(filepath,sizeof(gcf),6) != 0)
        {puts( btGetErr()); goto c9999;}
    if(bt_open(&fpgc,filepath) != 0)
        {puts( btGetErr()); goto c9999;}

    /* ------ ENDING PROGRAM ------ */
/* first find out the starting areas */
for ( i = 0, lw_count = 0; i <= MXU && strlen(obj.st_are[i]) > 6; i++) {
    if ( obj.st_are[i][6] == '3') /* is LW */
    {
        strcpy(tmpare[lw_count],obj.st_are[i]);
        lw_count ++;
    }
    else /* if DV, starting status always 'A' */
    {
        bt_read(fpqp, obj.st_are[i], &qpf);
        qpf.status = 'A';
        bt_rewrite(fpqp, qpf.areseq, &qpf);
    }
}

/* add pseudo LW to the QP file, lw_count records how many pseudo LWs */
for (i = 1; i <= lw_count+1; i++) {
    strcpy(qpf.areseq,tmpare[i-l]);
    strcpy( qpf.arenxt,qpf.areseq);
    strcpy(qpf.areseq, "0099993"); /* an initial pseudo LW area */
    qpf.status = 'A'; /* first pseudo LW always 'A' */
    qpf.aretype = ILl; /* it's a LW */
    for (j = 0; j < 4; j++)
    {
        qpf.mpx[j] = 0;
        qpf.mpy[j] = 0;
        qpf.qpara[j] = obj.qpara[j]; /* set pseudo LW qpara as required*/
    }
    for (j = 0; j < MXDV; j++)
        strcpy( qpf.adjare[j], "" );
    qpf.totton = 10; /* tons for pseudo LW, negligible */
    qpf.tadv = 0;
    qpf.pext = 100;

    scode(l,2,qpf.areseq,l,&i);
    bt_write(fpqp, qpf.areseq, &qpf);
}

/* begin to initialize the scheduling area list */
bt_start(fpqp, &qpf);
doi {
    if ( qpf.status == 'A')
    {
        strcpy(tmpare[0], qpf.areseq);
        aretemp = ini_area( tmpare[0]);
        if ( add_area(arelist, aretemp ) != 0 )
        { error(" Error in adding new area !"); goto c9999;}
        chk_status(arelisl,aretemp->seq,'M'); /* once started, status A->M*/
        bt_read(fpqp, tmpare[0], &qpf); /* restore the currency in qp file */
    }
} while ( bt_read_next (fpqp, &qpf) != 2 );

/* begin to initialize LiNDO */
i = 1;
llindo();
init();
l_tmp = -1;
quiet(&l_tmp);

/* Scheduling loop */
while( arelist->head != NULL && i < FLAG + 1 )
{
    ini_list(arelist, obj);
/* write LINDO MIP model */
wt_lindo( arelist, &obj);

/* execute LINDO */
lndio();
init();

/* read from LINDO output file */
rd_lindo( arelist, &obj);

/* write GC file */
wt_gc(arelist,&obj);
i++;

} /* Close all file handles and free memory */
c9999:
if( arelist->head != NULL )
{
    for( areptr = arelist->head, aretmp = areptr->next ;
        areptr != NULL; aretmp = areptr->next )
    {
        free(areptr);
        areptr = aretmp;
    }
}
bt_close( fpqp);
btclose( fpgc);
fclose(rel_ptr);
_setvideomode(_DEFAULTMODE);
return;

/*-------------------------------------------------------------------*
Function that initializes an area list at the beginning of each
scheduling period. Parameters of an area is checked by chk_para()
Return: void
*-------------------------------------------------------------------*/
void ini_list( ARELIST *arelist, OBJECT obj)
{
    AREA *areptr,*aretmp, *aretmpl;
    long flg;
    long i,j;
    char cc;
    /*
    At the beginning of each period, check whether there is any area
    going to be DONE. Also check quality parameters.
    In the list, All areas including possible successors are scheduled.
    If flg = 1 then an area is going to be completed in current period.
    */
    for (areptr = arelist->head; areptr != NULL; areptr = areptr->next) 
    {
        if(areptr->type == 'L') /* for LW */
        {
            areptr->rtwons = obj.tons_lw;
            flg = 0;
            if ( strcmp(areptr->seq + 2,"99993") == 0 ) /* pseudo LW only */
            {
                if ( areptr->pext == 0 ) continue; /* already checked */
            else
                areptr->pext = 0; /* borrow pext as a flg used
                only for pseudo LW */
            areptr->mtons = 10;
for(i = 0; i < 4; i++)
/* assign required qpara to pseudo LW */
    areptr->qpara[i] = obj.qpara[i];
    flg = 1;
}
else if( areptr->ctons - areptr->ptons > MINTON )
{
    /* if area was mined last period
    Check parameters. in: rons. out: mtons.
    1: status = 'M', mtons = rtons, flg = 0.
    2: status = 'D', mtons = remaining tons.
    flg = 1, re-check later for its next areas */
    chk_para(areptr,&flg);
}
else /* if was not mined last period */
{
    /* Area was waiting last period.
    'A': its precessor is still uncompleted.
    'M': itself in waiting.
    'D': when waiting in 'D' status, mtons = remaining tons.
    Parameters unchanged this time */
    continue;
}
else /* for DV */
{
    areptr->rtons = obj.tons_cm;
    flg = 0;
    if( areptr->ctons - areptr->ptons > MINTON )
    /* if DV area was mined last period
    Check parameters. in: rons. out: mtons.
    1: status = 'M', mtons = rtons, flg = 0.
    2: status = 'D', mtons = remaining tons.
    flg = 1, re-check later for next areas */
    chk_para(areptr,&flg);
}
else /* if was not mined last period */
{
    if( areptr->status == 'M' && areptr->mtons < areptr->rtons )
    /* for DV areas that was turned to 'M' by its precessor,
    but was not mined immediately. Its mtons is no longer
    restricted to remaining tons of its precessor. Recheck
    is needed for starting a new period */
    areptr->cadv = areptr->padv; /*retore padv for rechecking */
    chk_para(areptr,&flg);
}
else
{
    /* Area was waiting last period.
    Conditions:
    'A': its precessor is still uncompleted.
    'M': itself in waiting.
    'D': when waiting in 'D' status, mtons = remaining tons.
    Parameters unchanged this time */
    continue;
}
}
if ( flg == 1 ) /* area is completing */
{
chk_status( arelist, areptr->seq, 'D');

if( areptr->type == 'L') /* LW */
{
    if( areptr->nxt[6] == '0') continue; /* Is last LW */
    aretmp = inlist(arelist,areptr->nxt); /* nxt LW in list? */
    if( aretemp == NULL ) /* new LW */
    {
        if( (aretmp = ini_area(areptr->nxt)) == NULL )
            puts(areptr->nxt);
            error(" Error in initialize a new are !");
    }
    /* check status of next LW's DVs */
    for( i = 0; aretmp->dv[i].seq[6] != '0' &&
        strlen( aretmp->dv[i].seq ) > 6 ; i++ )
    {
        if((aretmp1 = inlist(arelist,aretmp->dv[i].seq)) != NULL )
            aretmp->dv[i].status = aretmp1->status;
    }
    /* rtons = remaining tons for new area */
    aretmp->rtons = areptr->rtons - areptr->mtons;
    /* check para here */
    flg = 0;
    chk_para(aretmp, &flg);
    if( flg == 1 ) /* no two areas can be done in the same period*/
        error( " The area is too small ");
    add_area(arelist,aretmp);
    chk_status(arelist,aretmp->seq, 'A'); /* next LW become A */
}
else /* already in list, re-check for updating */
{
    aretmp->rtons = areptr->rtons - areptr->mtons;
    /* check para here */
    flg = 0;
    chk_para(aretmp, &flg);
}
else /* when DV area is completing */
{
    /* open the entrance of all next DV areas */
    for(i=0; strlen(areptr->dv[i].seq) >1 &&
        areptr->dv[i].seq[6] != '0'; i++)
    {
        aretmp = inlist(arelist,areptr->dv[i].seq);
        if( aretemp == NULL )
            /* initialize DV */
            if( (aretmp = ini_area(areptr->dv[i].seq)) == NULL )
                puts(areptr->dv[i].seq);
                error(" Error in initialize a new are ");
            /* remaining tons is rtons for the succeeding area */
            aretmp->rtons = areptr->rtons - areptr->mtons;
            flg = 0;
            chk_para(aretmp, &flg);
            if( flg == 1 )
                error(" The area is too small ");
add_area(arelist, aretmp);
/* change new DV to 'A' */
chk_status(arelist, aretmp->seq, 'A');
}
else
{
/* remaining tons is rtons for the succeeding area */
aretmp->rtons = areptr->rtons - areptr->mtons;
flg = 0;
chk_para(aretmp, &flg);
}
}
/* end of for(;;) */
if (DEBUG == 0)
{
  for (areptr = arelist->head; areptr != NULL; areptr = areptr->next)
  {
    printf("area %s st %c tadv %8.2lf cadv %8.2lf
cady%8.0lf
", areptr->seq, areptr->status, areptr->tadv, areptr->cadv,
        areptr->padv, areptr->mtons);
  }
  puts("Want STOP ?");
  cc = getch();
  if (cc == 'y' || cc == 'Y')
    {
      _setvideomode(_DEFAULTMODE);
      exit(0);
    }
}
/*------------------------------------------------------------------------------------------*/
/* Function that adds to the list (always to the tail) a new area that has been initialized by ini_area(); */
static long add_area(ARELIST *arelist, AREA *area)
{
/* add an area at the end of a list */
if (area == NULL)
{
  error("Add an empty area !");
  return 1;
}
if (arelist->head == NULL) /* when list is empty */
{
  arelist->head = area;
  arelist->tail = area;
  return 0;
}
else
{
  (arelist->tail)->next = area; /* old tail area connects to new area */
  arelist->tail = area; /* new tail points to new area */
  return 0;
}
static AREA *ini_area( char areseq[] )
{
    AREA *area;
    long i;
    struct qptype qpf;

    if( strlen(areseq) < 7 || areseq[6] == '0') return NULL;/* not valid name*/;
    if ( ( area = malloc( sizeof(AREA) ) ) == NULL )
    {
        puts ( "\n\n\n RUN OUT OF MEMORY IN THE QUALITY SCHEDULING !\n" "PLEASE CONTACT WITH YOUR SYSTEM MANAGER TO SOLVE
THE PROBLEM \n" );
        wait();
        return NULL;
    }

    strcpy( area->seq, areseq);
    if( bt_read(fpqp,area->seq,&qpf) != 0 ){error("Can not find this area in QP file!"); return NULL;}
    /* initialize variables */
    strcpy(area->nxt,qpf.arenxt);
    strcpy(area->lndnm,"");
    for( i = 0; i < 4; i++)
    {
        area->mpx[i] = qpf.mpx[i];
        area->mpy[i] = qpf.mpy[i];
        area->px[i]  = 0;
        area->py[i]  = 0;
        area->qpara[i] = 0;
    }

    area->ttons = qpf.totton;
    area->rtons = 0;
    area->ctons = 0;
    area->ptons = 0;
    area->mtons = 0;
    area->tadv  = qpf.tadv;
    area->padv  = 0;
    area->cadv  = 0;
    area->pext  = qpf.pext;

    if ( ( area->seq[6] == '3' ) )
    {
        area->type = 'L';
        area->status = '"';

        /* read from QP file for its next LW surrounding areas and must check for their tonnage and status information. 
         * Read info of next LW and save to qpf */
        bt_read(fpqp,area->nxt,&qpf);
        for( i = 0; i < MXDV; i++)
            strcpy( (area->dv[i]).seq,qpf.adjare[i] );
        for( i = 0; area->dv[i].seq[6] != '0' &&
            strlen( area->dv[i].seq ) > 6; i++)
        {
            bt_read(fpqp,area->dv[i].seq,&qpf);
            area->dv[i].status = qpf.status;
            area->dv[i].ttons = qpf.totton;
        }
    }
    else
    {
        area->type = 'C';
        area->status = '"';

        /* Read from QP file the DV possible next areas. 
         * No need to check other information */
        for( i = 0; i < MXDV; i++)
strcpy(area->dv[i].seq,qpf.adjare[i]);
area->dv[i].status = 'U';
}
area->next = NULL; /* New area always points to NULL */
return area;

/*-------------------------------------------------------------------*
| Function that generates error messages. |
| Return void. |
*-------------------------------------------------------------------*/
void error( char msg[] )
{
    puts(msg);
    wait();
    _setvideomode(_DEFAULTMODE);
    exit(1);
}

/*-------------------------------------------------------------------*
| Function that finds out a given area is in the list or not |
| Return: current position in the list if found, NULL otherwise. |
*-------------------------------------------------------------------*/
AREA *inlist(ARELIST *arelist, char areseq[])
{
    AREA *p;
    for(p = arelist->head; p != NULL ; p = p->next )
    {
        if( strcmp(areseq,p->seq) == 0 )
            return p;
    }
    return NULL;
}

#endif

#include "d:\prog11\quick\icampsll.h"
#include "d:\yang\thesis\qplan.h"

static int scpy(char *str1, int stl, char *str2, int st2, int len);

/*-----------------------------------------------------------------
| Function that open an objective.def file and read in scheduling |
| objectives. The OBJECT struct is defined in qplan.h |
*-----------------------------------------------------------------*/
void open_object( OBJECT * obj)
{
    FILE *fp_obj;
    long i;

    char buffer[100];
    char filepath[80];
    strcpy(FNAMECUR,""); // |

    /* creating and opening scheduling objective file */
    strncpy(filepath,FNAMECUR);
    strcat(filepath,"object.def");
    fp_obj = fopen(filepath,"r");

    /* read objective paremeters */
    fgets(buffer,80,fp_obj);
    for( i = 0; i < MXU ; i ++)
    {
scpy(obj->st_are[i],1,buffer,7*i+1,7);
if (strlen(obj->st_are[i]) < 7) break;
}
fscanf(fp_obj,"%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%lf%l
{
    scpy(qpf.areseq,1,buffer,1,7);
    scode(0, 1, qpf.areseq, 7, &areatype);
    if( areatype == 3 )
    {
        qpf.aretype = 'L';
        scpy(qpf.arenxt,1,buffer,8,7);
        i = 15; /* skip one field from 7 to 14 */
    }
    else
    {
        qpf.aretype = 'C';
        strcpy(qpf.arenxt,"");
        i = 8;
    }
    for( j = 0; j < MXDV; j++, i+=7)
    {
        if( scpy(qpf.adjare[j],1,buffer,i,7) != 0 ) break;
    }
    while( j < MXDV )
    {
        strcpy(qpf.adjare[j],"");
        j++;
    }
    if(bt_read(fpse,qpf.areseq,&sef) != 0)
    {
        puts( btGetErr());
        goto c9999;
    }
    qpf.status = 'U';
    for( i = 0; i < 4; i++ )
    {
        qpf.mpx[i] = sef.mpx[i];
        qpf.mpy[i] = sef.mpy[i];
    }
    /* calculating area middle line length as total length of an area */
    area_middle_line(qpf.mpx,qpf.mpy,&qpf.tadv);
    /* store pext: percentage of area under pillars */
    qpf.pext = sef.pext;
    /* create tempry area for calculation */
    areptr = &area;
    strcpy (areptr->seq,qpf.areseq);
    areptr->pext = qpf.pext;
    for(i=0;i<4;i++)
    {
        areptr->mpx[i] = qpf.mpx[i];
        areptr->mpy[i] = qpf.mpy[i];
        areptr->px[i] = qpf.mpx[i];
        areptr->py[i] = qpf.mpy[i];
    }
    /* calculating total tonnage of an area */
    scode(0, 1, areptr->seq, 7, &areatype);
    scode(0, 4, areptr->seq, 3, &areanum);
    /* Interface ICAMPS for grid info.*/
    prop_area_grid(areptr->px, areptr->py, areanum, areatype,
        igrid, idiff, areatype);
    /* get average quality parameters from globle variables */
    qpf.qpara[0] = TPARAM[4]; /* sulfur */
    qpf.qpara[1] = TPARAM[5]; /* ash */
    qpf.qpara[3] = TPARAM[7]; /* moisture */
    calc_tons(areptr);
    qpf.totton = areptr->mtons;
}
if (bt_write(fpqp, qpf.aresseq, &qpf) != 0) {
    puts(btGetErr()); goto c9999;
} /* end of while() */

/* close all input files */
bt_close(fpse);
bpt_close(fpqy);
fclose(rel_ptr);
return;

c9999:
    error("Error in routine open_qp() !!! ");
    bt_close(fpqp);
    fclose(rel_ptr);
    return;
}

static int scpy(char *str1, int stl, char *str2, int st2, int len)
{
    /* function string copy:
       It copies characters from string 2 starting at st2 to
       string 1 starting from stl. String 1 may not be initialized.
       If normal terminated, string 1 has the length of stl+len.
       and return value 0; otherwise return 1 */
    int i;
stl--;st2--;
i=0;
while( i < stl) { strl[i]=' '; i++; }
for(i=0;i<len;i++)
{
    strl[stl+i]=str2[st2+i];
    if(str2[st2+i] == '\0') return 1;
}
strl[stl+len]='\0';
return 0;
}

WT_LINDO.C

#include "d:\prog1\quick\icamps11.h"
#include "d:\yang\thesis\qplan.h"

/* structure that represent a row */
typedef struct tag_ROW {
    char name[9];
    char sign; /* one of E: =, L: <=, G: >= */
    double rhs;
} ROW;

/* struct that represent a variable in a linked list
   following the order of occurance */
typedef struct tag_VAR {
    char name[9]; /* var name */
    char row_nm[9]; /* in which row */
    double value; /* coefficient */
    struct tag_VAR *next;
} VAR;

typedef struct tag_VARLIST {
    VAR* head;
} VARLIST;

char *l_a( long value); /* function that convert a long to a string */
VAR* creat_var(void);
void insert_var(VARLIST * list_head ,VAR * var);
void def_row( ROW *row, char name[], char sign, double rhs);
VAR *def_var( char name[], char row_nm[], double value);

void mkname(AREA *area); /* function that make Indio name */

/* Function that writes for LINDO the input file in MPS format */

void wt_lindo( ARELIST *arelist, OBJECT *obj)
{
    static long ff = 0;
    char name[9],*tmp;
    FILE *fp_lnd_in;
    ROW row[100];
    VAR *var_ptr, *var_tmp;
    VARLIST i_list;
    VARLIST f_list;
    AREA *areptr, *aretmp;
    char filepath[80];
    char intg[50][4]; /* integer var name */
    char remlw[5][8]; /* remaining LW tonnage name */
    char remdv[5][8]; /* remaining DV tonnage name */
    char slk[5][8];
    char line_name[6][6] = ("SULF", "ASH", "BTU", "MOIS", "TONL", "TONC");
    char p_dev[6][8]; /* positive deviation name */
    char n_dev[6][8]; /* negative deviation name */
    char tmpsl[80], tmps2[80]; /* temp strings for writing constraints*/
    long i, row_count, row_save, int_count, lw_count;
    double total_tons; /* tmp total tons */
    double rhs; /* tmp right hand side */

    /*-------------------- create input file for lindo ------------------------*/
    /* creating and opening linda input roadIe file */
    strcpy(name, "LNDIN");
    ff++;
    tmp = l_a(ff);
    strcat(name,tmp);
    rename ("lndin", name);

    strcpy(filepath,FNAMECUR);
    strcat(filepath, "LNDIN");
    remove(filepath);
    fp_lnd_in = fopen (filepath, "w");

    /********* initialize float_list and integer_list *********/
    f_list.head = NULL;
    i_list.head = NULL;

    /********* initializing variable names *********/
    for ( i=0; i<6; i ++ )
    {
        strcpy( p_dev[i], "PDEV");
        strcat( p_dev[i], l_a(i + 1));
        strcpy( n_dev[i], "NDEV");
        strcat( n_dev[i], l_a(i + 1));
    }

    /* initialize integer variables */
    for( i = 0; i<50 ; i ++ )
    {
        strcpy ( intg[i], "Y");
        strcat ( intg[i], l_a(i));
    }

    /* initialize remaining tonnage variables */
```c
for( i = 0; i< 5 ; i ++ )
{
    strcpy (remlw[i],"RMLW");
    strcat (remlw[i],_a(i+l) );
    strcpy (remdv[i],"RMDV");
    strcat (remdv[i],_a(i+l) );
    strcpy (slk[i],"SLK");
    strcat (slk[i],_a(i+l) );
}

lw_count = 0;
tottons = obj->ttens_lw + obj->ttens_cm;

/* assign each area with LINDO variable name */
for( areptr = arelist->head,i = 1; areptr != NULL; 
    areptr = areptr->next,i++)
    mkname(areptr);

/******** Begin writing objective function:
use has to specify to optimize which quality parameters
and the cost coefficients for each of them ******/

/* ------------------------------- MIN -------------------------------*/
row_count = 1;
def_row(&row[row_count - 1],l_a(row_count),'N',0.0);

/* store row #1 variables in var_list */
for ( i = 0; i < 6; i ++ )
{
    if ( obj->qpara[i] == 0 && i < 4 ) continue;
    var_tmp = def_var(p_dev[i],l_a(row_count),obj->p_wt[i]);
    insert_var(&f_list,var_tmp);
    var_tmp = def_var(n_dev[i],l_a(row_count), obj->n_wt[i]);
    insert_var(&f_list,var_tmp);
}

/*---------------------- SUBJECT TO ------------------------- */
/* Write primary tonnage and quality parameters constraints */
/* quality req. equations 1: sulfur 2: ash 3: btu 4: mois */
for ( i=0; i<4 ; i ++ )
{
    if ( obj->qpara[i] == 0 ) continue; /* if no this quality para */
    row_count ++;
def_row(&row[row_count - 1],line_name[i], 'E',0);
    for( areptr = arelist->head; areptr != NULL; 
        areptr = areptr->next)
    {
        /* if quality para. is not available, assigne average to it */
    }
```

if (areptr->qpara[i] < 0 ) areptr->qpara[i] = obj->qpara[i];
    var_tmp = def_var(areptr->lndnm,line_name[i],
    areptr->qpara[i]-obj->qpara[i]);
    insert_var(&f_list,var_tmp);
}

    var_tmp = def_var(n_dev[i],line_name[i],1);
    insert_var(&f_list,var_tmp);
    var_tmp = def_var(p_dev[i],line_name[i],-1);
    insert_var(&f_list,var_tmp);
}

/* tonnage requirement for all LW including possible successors */
row_count ++;
def_row(&row[row_count - 1],line_name[4],E,obj->ttons_lw);

for( areptr = arelist->head; areptr != NULL; areptr = areptr->next) {
    if ( areptr->type == 'L' )
    {
        var_tmp = def_var(areptr->lndnm,line_name[4],1);
        insert_var(&f_list,var_tmp);
    }
}

var_tmp = def_var(n_dev[4],line_name[4],1);
    insert_var(&f_list,var_tmp);
    var_tmp = def_var(p_dev[4],line_name[4],-1);
    insert_var(&f_list,var_tmp);
/* tonnage constraint for DV which controls how many crews working
on CM units, also including possible successors */
row_count ++;
def_row(&row[row_count - 1],line_name[5],E,obj->ttons_cm);

for( areptr = arelist->head; areptr != NULL; areptr = areptr->next) {
    if ( areptr->type == 'C' )
    {
        var_tmp = def_var(areptr->lndnm,line_name[5],1);
        insert_var(&f_list,var_tmp);
    }
}

var_tmp = def_var(n_dev[5],line_name[5],1);
    insert_var(&f_list,var_tmp);
    var_tmp = def_var(p_dev[5],line_name[5],-1);
    insert_var(&f_list,var_tmp);
/* For all dynamic constraints, check mining conditions
or each area in the scheduling list */

for( areptr = arelist->head; areptr != NULL; areptr = areptr->next) {
    if( areptr->status == 'A') continue; /* no constraints for 'A' */
    if( areptr->type == 'L')
    { /* boundary of LW */
        if( areptr->status == 'M') /* Mining going on */
        { /* areptr->lndnm <= areptr->mtons */
            row_count ++;
def_row(&row[row_count - 1],1_a(row_count),L,areptr->mtons);
            var_tmp = def_var(areptr->lndnm,1_a(row_count),1);
            insert_var(&f_list,var_tmp);
        } /* if last LW , no more constraints */
        else if( areptr->status == 'D') /* will be done in this period */
/* tons(current lw) <= remaining tons in the area */
row_count ++;
def_row(row[row_count -1],l_a(row_count),'L',areptr->mtons);
var_tmp = def_var(areptr->lndnm,l_a(row_count),1);
insert_var(&f_list,var_tmp);

/* if last LW, no more constraints */
if( areptr->nxt[6] == '0') continue;

/* tons(current lw) + Y1 * BIG_M >= remaining tons in the area
This constraint is used to control triggering next LW */
row_count ++;
def_row(row[row_count -1],l_a(row_count),'G',areptr->rtons);
var_tmp = def_var(areptr->lndnm,l_a(row_count),1);
insert_var(&f_list,var_tmp);

int_count ++;
var_tmp = def_var(intg[int_count],l_a(row_count),BIG_M);
insert_var(&i_list,var_tmp);

/* Y1 + Y2 <= 1 */
row_count ++;
def_row(row[row_count -1],l_a(row_count),'L', 1.0);
var_tmp = def_var(intg[int_count],l_a(row_count),1);
insert_var(&i_list,var_tmp);
int_count ++;
var_tmp = def_var(intg[int_count],l_a(row_count), 1);
insert_var(&i_list,var_tmp);

/* find out next long wall ID in the list */
for (aretmp=arelist->head; aretmp != NULL &&
strcmp(areptr->nxt,aretmp->seq) != 0; aretmp = aretmp->next);

/* tons (current lw) + tons(next lw) <= required tonnage */
row_count ++;
def_row(row[row_count -1],l_a(row_count),'L', areptr->rtons);
var_tmp = def_var(areptr->lndnm,l_a(row_count),1);
insert_var(&f_list,var_tmp);
var_tmp = def_var(aretmp->lndnm,l_a(row_count),1);
insert_var(&f_list,var_tmp);

/* tons( next lw ) <= Y2 * difference between rtons and mtons */
row_count ++;
def_row(row[row_count -1],l_a(row_count),'L',0.0 );
var_tmp = def_var(areptr->lndnm,l_a(row_count),1);
var_tmp = def_var(intg[int_count],l_a(row_count), (areptr->rtons-areptr->mtons));
insert_var(&i_list,var_tmp);

/*----------------- look ahead constraints ------------------------- */
/* remaining tons of LW */
/* areptr->lndnm + remlw[lw_count] = areptr->ttons - areptr->ctons */
row_count ++;
def_row(row[row_count -1],l_a(row_count),'
E',areptr->ttons - areptr->ctons);
var_tmp = def_var(areptr->lndnm,l_a(row_count), 1);
insert_var(&f_list,var_tmp);
var_tmp = def_var(remlw[lw_count],l_a(row_count),1);
insert_var(&f_list,var_tmp);

/* remaining tons of its next LW surrounding DV areas */
rhs = 0;
row_count ++;
for ( i = 0; strlen(areptr->dv[i].seq) > 6 &&
    areptr->dv[i].seq[6] != '0'; i ++ )
{
    /* area has not been mined */
    if ( areptr->dv[i].status == 'U')
    {
        rhs += areptr->dv[i].ttons;
    }
    else /* whether it is M or A or D */
    {
        for (aretmp=arelist->head;
            strcmp(areptr->dv[i].seq, aretmp->seq)!= 0 &&
            aretmp != NULL; aretmp = aretmp->next);
        if( aretmp == NULL) continue; /* DV[i] Done, not in list */
        rhs += aretmp->ttons - aretmp->ctons;
        var_tmp = def_var(aretmp->lndnm,l_a(row_count), 1 );
        insert_var( &f_list,var_tmp);
    } /* end of for(;;) */
}

/* area has not been mined */
if ( areptr->dv[i].status == 'U')
{
    rhs += areptr->dv[i].ttons;
}
else /* whether it is M or A or D */
{
    for (aretmp=arelist->head;
        strcmp(areptr->dv[i].seq, aretmp->seq)!= 0 &&
        aretmp != NULL; aretmp = aretmp->next);
    if( aretmp == NULL) continue; /* DV[i] Done, not in list */
    rhs += aretmp->ttons - aretmp->ctons;
    var_tmp = def_var(aretmp->lndnm,l_a(row_count), 1 );
    insert_var( &f_list,var_tmp);
}
} /* end of for(;;) */
/* DV + BIG_M * Y1 >= areptr->mtons (Remaining tons)
which means that unless DV = Remaining tons, then Y can be
set to 0, otherwise, it is always 1. If it is set to 1,
then all the successors will not be started. */

row_count ++;
def_row(&row[row_count - 1], l_a(row_count), 'G', areptr->mtons);
var_tmp = def_var(areptr->lndnm, l_a(row_count), 1);
insert_var(&f_Ilist, var_tmp);
int_count ++;
var_tmp = def_var(intg[int_count], l_a(row_count), BIG_M);
insert_var(&i_list, var_tmp);

/* Y1 + Y2 <= 1 which controls only one of integers can be 1 */
row_count ++;
def_row(&row[row_count - 1], l_a(row_count), 'L', 1);
var_tmp = def_var(intg[int_count], l_a(row_count), 1);
insert_var(&i_list, var_tmp);
int_count ++;
var_tmp = def_var(intg[int_count], l_a(row_count), 1);
insert_var(&i_list, var_tmp);

/* put all successors of this DV into competitive scheduling */
for ( i = 0; strlen(areptr->dv[i].seq) > 6 &&
         areptr->dv[i].seq[6] != '0' ; i ++ )
{
    if( strcmp(areptr->dv[i].seq, aretmp->seq) == 0 &&
        aretmp->status == 'A')
    {
        /* conditional boundary of new DV area */
        /* aretmp->lndnm - (areptr->rtons - areptr->mtons)Y2 <= 0*/
        row_count ++;
def_row(&row[row_count - 1], l_a(row_count), 'L', 0.0);
        var_tmp = def_var(aretmp->lndnm, l_a(row_count), 1);
        insert_var(&i_list, var_tmp);
        var_tmp = def_var(intg[int_count], l_a(row_count), -(areptr->rtons-areptr->mtons));
        insert_var(&i_list, var_tmp);
        break; /* since found that area in the list so
        break this for(;;) to search next one */
    }
}

/* Write the LNDIN file in MPS format */printf ( fp_lnd_in, "NAME
ROWS
" );
for ( i = 0; i < row_count; i ++)
    fprintf ( fp_lnd_in, "@ %s
", row[i].sign, row[i].name);
for( int_count != 0 )
{
    fprintf ( fp_lnd_in,
        " @ INTEGER1 'MARKER'
        'INTORG' 
"
    );
    for( var_tmp = i_list.head; var_tmp != NULL; var_tmp = var_tmp->next )
    {
        fprintf ( fp_lnd_in, " @-10s%-10s%-1f
",
            var_tmp->name, var_tmp->row_nm, var_tmp->value);
    }
    fprintf ( fp_lnd_in,
        " @ INTEGER2 'MARKER' 'INTEND'
"
    );
}
for( var_tmp = f_list.head; var_tmp != NULL; var_tmp = var_tmp->next )
  { 
    if( var_tmp->value == 0 ) continue;
    fprintf ( fp_lnd_in," %-10s%-10s%lf\n",
      var_tmp->name,var_tmp->row_nm,var_tmp->value);
  }

fprintf ( fp_lnd_in,"RHS\n");
for( i = 1; i < row_count; i++)
  { 
    if( row[i].rhs == 0 ) continue;
    fprintf (fp_lnd_in, " RHS %10s%lf\n",row[i].name,row[i].rhs);
  }

if( int_count != 0 )
  {
    fprintf ( fp_lnd_in,"BOUNDS\n");
    for( i = 0; i < int_count; i++)
      fprintf ( fp_lnd_in," UP LINDO BND %10s%lf\n", intg[i+1], 1.0);
  }

fprintf ( fp_lnd_in,"ENDATA\n");

/* free memory */
if ( f_list.head != NULL )
  {
    for( var_ptr = f_list.head, var_tmp = var_ptr->next ;
      var_ptr != NULL; var_tmp = var_ptr->next )
      { free(var_ptr);
        var_ptr = var_tmp;
      }
  }

if( i_list.head != NULL )
  {
    for( var_ptr = i_list.head, var_tmp = var_ptr->next ;
      var_ptr != NULL; var_tmp = var_ptr->next )
      { free(var_ptr);
        var_ptr = var_tmp;
      }
  }
fclose(fp_lnd_in);
return;

/*------------------------------------------------------
Function that creates a LINDO variable
-----------------------------------------------------*/
VAR * creat_var(void)
{
  VAR *p;
  if ( (p = (VAR *) malloc ( sizeof (VAR) ) ) == NULL )
    error (" No enough memory !!");
  return p;
}

/*------------------------------------------------------
Function that inserts a variable
 1. at the end of the whole list, if its own family not in list,
 2. or at the end of the sublist of its own family.
------------------------------------------------------*/
void insert_var ( VARLIST * list, VAR * var)
{
  VAR *p, *q;
  if ( list->head == NULL ) /* if empty list */
    {
      list->head = var;
  }
for (p = list->head; p->next != NULL; p = p->next)
{
    if (strcmp(p->name, var->name) != 0) continue;
    else
    {
        if (strcmp((p->next)->name, var->name) == 0)
            continue;
        q = p->next;
        p->next = var;
        var->next = q;
        return;
    }
}
/* for last element in the list */
p->next = var;
var->next = NULL;
return;
*/

Function that defines a Row in LINDO MPS format input file
---------------------------------------------------------------
void def_row(ROW *row, char name[], char sign, double rhs)
{
    strcpy(row->name, name);
    row->sign = sign;
    row->rhs = rhs;
    return;
}

Function that defines a variable in LINDO MPS format input file
----------------------------------------------------------------
VAR * def_var( char name[], char nm[], double value)
{
    VAR * tmp;
    tmp = creat_var();
    strcpy(tmp->name, name);
    strcpy(tmp->row_nm, nm);
    tmp->value = value;
    return tmp;
}

Function that makes a NULL terminated lindo name for an area
---------------------------------------------------------------
void mkname(AREA *area)
{
    long sub, id;
    scode(0,4,area->seq,3,&id);
    scode(0,2,area->seq,1,&sub);
    if(area->type == 'L')
        strcpy(area->lndnm, "LW");
    else
        strcpy(area->lndnm, "DV");
    strcat(area->lndnm, l_a(id));
    if( sub != 0 )
    {
        strcat(area->lndnm, "_");
        strcat(area->lndnm, l_a(sub));
    }
    return;
}
/* Function char * l_a( long value) 
It changes a long to a NULL terminated string */
char *l_a( long value) 
{
static char tmpstr[10];
char *tmp;
tmp = tmpstr;
ltoa(value,tmp,10);
return tmp;
}

#include "d:\progll\quick\icampsll.h"
#include "d:\yang\thesis\qplan.h"

RD_LINDO.C

/* Function that reads the output from LINDO DBS format report 
and stores tonnage assignment for each unit and the final 
quality of that area. 
Variables may be updated in this routine:
area->mtons: gets actual mined tons after scheduling 
obj->q_dev: all quality parameters' deviations 
if the model is correct, integers and other variables 
are not in interest. */

void rd_lindo( ARELIST *arelist, OBJECT *obj )
{
FILE *fp_lnd_out;
AREA *areptr;
char buffer[80];
char filepath[80];
char varnm[50][9]; /* variable name in report */
double value[50]; /* variable value in report */
double tmptons;
long i,var_count, flg, j;

/******** opening lindo output file ********/
strcpy(filepath,FNAMECUR);
strcat(filepath,"LNDOUTl");
fp_lnd_out = fopen(filepath,"r");

/******** read relation file to find area-relation of this area ********/
fgets(buffer,80,fp_lnd_out); /* skip the objective line */
i = 0;
while( fgets(buffer,80,fp_lnd_out) != NULL )
{
sscanf(buffer, "%s%lg", varnm[i],&value[i]);
if(varnm[i][0] == 'Y' || varnm[i][0] == 'R') continue;
i++;
}
var_count = i;

/* store the assigned tonnage by LINDO to corresponding unit 
if actual tons assigned is less than the required tons 
an adjustment is needed for qpara[] */
for ( areptr = arelist->head; areptr != NULL ; areptr = areptr->next )
{
/* locate the mining unit corresponding to vname. i is only 
determined by this for(;;) and should not be changed later*/
for ( i = 0; i < var_count && 
strcmp(areptr->lndnm,varnm[i]) != 0; i ++ );
}
if( value[i] <= MINTON ) /* area was actually not mined */
{
    areptr->ptons = areptr->ctons;
    areptr->ctons += 0;
}
else if( value[i] > MINTON ) /* actually mined something */
{
    if( areptr->mtons - value[i] > MINTON )
    {
        /* area was not mined as required, but some job has been
         done so the parameters of this smaller portion needs to
         be re-checked for accuracy */
    tmptons = areptr->rtons; /* save rtons */
    areptr->rtons = value[i]; /* The value[] passed in chk_para()
     will force it to assign that
     much tonnage to value[] */
    areptr->ctons += value[i]; /* The value[] passed in chk_para()
     will force it to assign that
     much tonnage to value[] */
    areptr->rtons = tmptons; /* restore rtons */
    if( fabs( areptr->ttons - areptr->ctons) < MINTON )
    {
        areptr->ttons = 0; /* a mark for a completed area */
    }
}
/* store the quality and tonnage parameters deviation */
for( i = 0; i < var_count; i++)
{
    if( varnm[i][4] == '1')
    {
        if( value[i] != OL )
            obj->q_dev[0] = value[i];
        else
            obj->q_dev[0] = -value[+i];
    }
    else if ( varnm[i][4] == '2')
    {
        if( value[i] != OL )
            obj->q_dev[1] = value[i];
        else
            obj->q_dev[1] = -value[+i];
    }
    else if ( varnm[i][4] == '3')
    {
        if( value[i] != OL )
            obj->q_dev[2] = value[i];
        else
            obj->q_dev[2] = -value[+i];
    }
    else if ( varnm[i][4] == '4')
    {
        if( value[i] != OL )
            obj->q_dev[3] = value[i];
    }
else
    obj->q_dev[3] = -value[++i];
else
{
    continue;
}
}
fclose( fp_lnd_out);
return;

WT_GC.C

#include "d:\progll\quick\icampsll.h"
#include "d:\yang\thesis\qplan.h"

/* -----------------------------------------------------------------------
Function that writes final report to GC file (Gantt Chart type). 
Area mining status is updated according to the scheduling result in the 
last period. Also mining status of an area's next area(s) is updated 
when it was completed. Also the QP file is updated. 
-----------------------------------------------------------------------*/

void wt_gc(ARELIST *arelist, OBJECT *obj)
{
    static long record = 0;
    AREA *areptr,*aretmp;
    double tttons_lw = 0, tttons_cm = 0;
    double qdev[4],deviation;
    struct gctype gcf;
    struct qptype qpf;
    long i,j,count=0,tmp;
    extern void _draw_poly_w( double x[],double y[],short vertex, short color,
    short fill);
    void chk_status(ARELIST *arelist, char areseq[], char new_status);
    static long del_area(ARELIST *arelist,AREA *area);
    strcpy(gcf.pd,"");
    /* Write starting line of a period */
    strcpy(gcf.areseq, "start");
    record ++;
    tmp = record*100 + count;
    count++;
    scode(l,6,gcf.pd,l,&tmp) ;
    gcf.aretype = 'S';
    gcf.rtons = obj->ttons_lw; /* total tonnage required for LW */
    gcf.mtons = obj->ttons_cm; /* total tonnage required for CM */
    for( i = 0; i < 4; i ++ )
    {
        gcf.qpara[i] = obj->qpara[i]; /* quality para. req. %/ton */
        gcf.cost[i] = 0;
        qdev[i]= 0;/* initialize quality deviation */
    }

    if ( bt_write(fpgc,gcf.pd,&gcf) != 0)
        puts( btGetErr() );
    /* Write mining info. for each mining unit */
    areptr = arelist->head;
    while( areptr != NULL )
    {
        /* Nothing has done with this area , status unchanged*/
        if( (areptr->ctons - areptr->ptons) < MINTON )
        {
            areptr = areptr->next;
            continue;
        }
}
/* mined something, go on */
strcpy(gcf.areaseq,areptr->seq);
tmp = record*100 + count;
count++;
scode(1,6,gcf.pd,1,&tmp);
gcf.aretype = areptr->type;
gcf.rtons = areptr->rtons;
gcf.mtons = areptr->ctons - areptr->ptons;
if(gcf.aretype == 'L')
{
ttons_lw += gcf.mtons;
_draw_poly_w( areptr->px,areptr->py,4, 9, 4);
}
else /* 'C' */
{
ttons_cm += gcf.mtons;
_draw_poly_w( areptr->px,areptr->py,4, 9, 3);
}
for( i = 0; i < 4; i ++ )
{
gcf.qpara[i] = areptr->qpara[i]; /*area avg. quality in period */
deviation = areptr->qpara[i] - obj->qpara[i];
qdev[i] += deviation * gcf.mtons; /* substance contained in dev.*/
gcf.cost[i] = ( deviation > 0 ) ?
( deviation * obj->p_wt[i] * gcf.mtons ) :
((0.0 - deviation) * obj->n_wt[i] * gcf.mtons );
}
/* when an area is competely mined */
if( areptr->status == 'D'&& areptr->ttons < MINTON )
{
/* If a LW area is completed, next LW will be 'M' */
if ( areptr->type == 'L')
{
/* This LW 'D', next LW 'M' */
chk_status(arelist,areptr->seq, 'D');
chk_status(arelist,areptr->nxt, 'M');
}
/* If a DV area is completed, updating its successor's status */
else
{
/* This DV 'D' */
chk_status(arelist,areptr->seq, 'D');
/* all next DV 'M' */
for( i = 0; strlen(areptr->dv[i].seq) > 6 &&
    areptr->dv[i].seq[6] != '0'; i ++)
chk_status(arelist,areptr->dv[i].seq, 'M');
}
/* change its mining status to "D" in QP file */
if ( bt_read(fpqp,areptr->seq,&qpf) != 0 )
puts( btGetErr() );
quf.status = 'D';
if ( bt_rewrite(fpqp,areptr->seq,&qpf) != 0 )
pus( btGetErr() );
/* write info about this deleted area in GC */
if ( bt_write( fpgc, gcf.pd, &gcf)!= 0 )
pus( btGetErr() );
/* !!! important list manipulation to avoid breaking list
because of deleting a node from the list */
aretmp = areptr;
areptr = areptr->nexti
/* Delete this area from the mining list */
if( del_area( arelist,aretmp ) != 0 )
error(" Delete area error ");
if( areptr == NULL ) goto end; /* last node in list */
else continue;
}
else /* area is not completed */
{ /* write info and proceed to next node */
  if ( bt_write( fpgc, gcf.pd, &gcf)!= 0 )
    puts( btGetErr() );
  areptr = areptr->next;
  if(areptr == NULL) goto end; /* if last node*/
  else continue;
}
end:/* Write end of a period statement */
strcpy(gcf.areseq,"ending");
tmp = record*100 + count;
count++; scode(1,6,gcf.pd,1,&tmp);
gcf.aretype = 'E';
gcf.rtons = ttons_lw; /* total tonnage mined by LW */
gcf.mtons = ttons_cm; /* total tonnage mined by CM */
for( i = 0; i < 4; i ++ )
{  gcf.qpara[i] = qdev[i]/( ttons_lw + ttons_cm); /* actual percentage of deviation from req. */
    gcf.cost[i] = ( gcf.qpara[i] > 0 ) ?
        ( gcf.qpara[i] * obj->p_wt[i]* (ttons_lw + ttons_cm)):
        ( ( 0.0 - gcf.qpara[i] ) * obj->n_wt[i]* (ttons_lw + ttons_cm));
}
if( bt_write(fpgc,gcf.pd,&gcf) != 0 )
  puts( btGetErr() );
  return;

/*-----------------------------------~------------------------------------------
Function that deletes an area from list and free memery.
--------------------------------------------------------------------------*/
static long del_area( ARELIST *arelist,AREA *area)
{
  AREA *p;
  if ( area == NULL )
    { error(" Try to delete an empty area ! "); return 1; }
  if ( area == arelist->head ) /* if the area is the head */
    {
      if( area == arelist->tail) /* only one area in list */
        {
          arelist->head = NULL;
          arelist->tail = NULL;
        }else
          arelist->head = area->next; /* remove the head */
        free(area);
        return 0;
    } /* search the area in list with at least two areas */
  p = arelist->head;
  while ( p->next != area)
    {
      p = p->next;
      if( p->next == NULL ) return 2; /* area not in list */
  p = p->next;
  if( p->next == NULL) goto end; /* if last node*/
/* found p->next == area, drop area out of list */
p->next = area->next;

/* if area is the last area in the list? */
if ( area->next == NULL ) arelist->tail = p;
free( area );
return 0;

/* ----------------------------------------------------------------------
   Function that changes the status of an area in the mining list
   ----------------------------------------------------------------------*/

void chk_status( ARELIST *arelist, char areseq[], char new_status)
{
    AREA *areptr, *aretmp;
    long i;
    if( strcmp(areseq, "0000000") == 0) return;
    if( areseq[6] == '3' ) /* LW */
    {  
        for( areptr = arelist->head; areptr != NULL; areptr = areptr->next )
            if(strcmp( areseq, areptr->seq) == 0 )
                { areptr->status = new_status;
                return;
        }
        printf("%s", areseq);
        error("Area not in the list");
    }
    else /* DV */
    {  
        for( areptr = arelist->head; areptr != NULL; areptr = areptr->next )
            if( strcmp( areseq, areptr->seq) == 0 )
            { areptr->status = new_status;
                for( aretmp = arelist->head; aretmp != NULL; aretmp = aretmp->next )
                    { 
                        for( i = 0; strlen(aretmp->dv[i].seq) > 6 ; i ++)
                            if( strcmp(areseq, aretmp->dv[i].seq) == 0 )
                                { 
                                    aretmp->dv[i].status = new_status;
                                    break;
                                }
                        }
                }
        printf("%s", areseq);
        error("Area not in the list");
    }
    return;
}

GRAF.C

/* Function that demo the mining process on the screen.
   Not absolutely needed for QCSMO */
#include "d:\progl\quick\icamps11.h"
#include "d:\yang\thesis\qplan.h"
#include<graph.h> 
#include<conio.h>
```c
#ifndef DEBUG
#define DEBUG 0
#endif

#define TRUE 1
#define FALSE 0
#define ON 1
#define OFF 0
#define PI 3.141569

int MaxX, MaxY;
int MaxColors;
void _draw_poly_w( double x[], double y[], short vertex, short color, short fill);
void graf(void)
{
  struct qptype qpf;
  int fpqp;
  int i;
  int mpx, mpy, px1, py1, px2, py2;
  double x[4], y[4];
  if ( _setvideomode( _VRES16COLOR) == 0)
  {
    printf(" Graphics System Error %s
");
    printf("Press any key to halt\n");
    getch();
    exit(1);
  }
  _setcolor( 15);
  mpx = 640-1;
  mpy = 480-1;
  _moveto(0,0);
  _lineto(mpx,0);
  _lineto(mpx,mpy);
  _lineto(0,mpy);
  _setwindow(1,63500.0,372000.0,87500.0,381000.0);
  /* open QP FILE for plotting the mine */
  if(bt_open(&fpqp, "work.qp") !=0)
  {
    _outtext("BTREE file work.qp open error\n "); exit(1); }
  _grtext_w(63500,380050, "QCAMO SCHEDULING");
  bt_start(fpqp,&qpf);
  do{
    _draw_poly_w(qpf.mpx,qpf.mpy,4,14,-1); 
    px+=x[0];
    py+=y[0];
  } while( bt_read_next(fpqp,&qpf) != 2 );
  bt_close(fpqp);
  return;
}

void _draw_poly_w( double x[], double y[], short vertex, short color, short fill)
{
  short i;
  double px=0, py=0;
  _setcolor(color);
  _moveto_w(x[0],y[0]);
  for ( i = 1; i < vertex ; i ++)
  {
    _lineto_w(x[i],y[i]);
    px += x[i];
    py += y[i];
  }
  _lineto_w(x[0],y[0]);
  px += x[0];
```

py += y[0];
if( fill >= 0 )
{
    _setcolor(fill);
    _floodfill_w( px/vertex, py/vertex, color );
}
return;

LOOK.C

/* Function that converts QP and GC file from BTREE file format to ASCII file format. Not absolutely needed for QCSMO */

#include "d:\prog11\quick\icamps11.h"
#include "d:\yang\thesis\qp1an.h"
void main(int argc,char *argv[])
{
    int fpqc,fpqp,tt;
    FILE *fp_gc,*fp_qp;
    struct gctype gcf;
    struct qptype qpf;
    char buffer[200];
    /*converting gcfile t ASCII*/
    if( strcmp(argv[1],"GC") == 0 || strcmp(argv[1],"gc") == 0 )
    {
        fp_gc = fopen("gc","w");
        if(bt_open(&fpqc,"work.gc")!=0)
        {
            puts("BTREE file work.gc open error\n "); exit(1); };
        bt_start(fpqc,&gcf);
        c5: puts(gcf.pd);
        sprintf(buffer,"%6s %7s %e %8.01f %8.01f %8.21f %8.21f %8.21f %8.21f
",
            gcf.pd,gef.areseq,gcf.aretype,gcf.rtons,gef.mtons,
            gcf.qpara[0],gef.qpara[1],gcf.cost[0],gcf.cost[1]);
        fprintf(fp_gc,"%s ",buffer);
        if( (tt=bt_read_next(fpqc,&gcf)) !=2 ) goto c5;
        fclose(fp_gc);
        bt_close(fpqc);
    }
    /*converting qpfile to ASCII*/
    if( strcmp(argv[1],"QP") == 0 || strcmp(argv[1],"qp") == 0 )
    {
        fp_qp = fopen("qp","w");
        if(bt_open(&fpqp,"work.qp")!=0)
        {
            puts("BTREE file work.qp open error\n "); exit(1); };
        bt_start(fpqp,&qpf);
        c10: sprintf(buffer,"%7s %c %c %7s %c %7s %c %7s %c %7s %c %7s %c %7s %c %7s %c %7s %c
",
            qpf.areseq,qpf.aretype,qpf.arenxt,qpf.status,
            qpf.totton,qpf.tadv,
            qpf.adjare[0],qpf.adjare[1],qpf.adjare[2],qpf.adjare[3],
            qpf.adjare[4],qpf.adjare[5],qpf.adjare[6],qpf.adjare[7],
            qpf.qpara[0],qpf.qpara[1],qpf.qpara[2],qpf.qpara[3],
            qpf.mpx[0],qpf.mpx[1],qpf.mpx[2],qpf.mpx[3],
            qpf.mpy[0],qpf.mpy[1],qpf.mpy[2],qpf.mpy[3],qpf.pext);
        fprintf(fp_qp,"%s ",buffer);
        if( (tt=bt_read_next(fpqp,&qpf)) !=2 ) goto c10;
        fclose(fp_qp);
        bt_close(fpqp);
    }
}
ICAMPS LIB and Functions are necessary for proper function of QCSMO. They provided basic file operations, grid file interpretation, tonnage calculations and other required geological operations. The following are LIBs and Files linked by QCSMO:

- **file** psobj.obj
- **file** qpobj.obj
- **file** calctons.obj
- **file** externs.obj
- **file** gridin.obj
- **file** tons.obj
- **file** dgeom.obj
- **file** geom.obj
- **file** coalgeom.obj
- **file** miscmath.obj
- **file** azimuth.obj
- **library** IC_SCRN.LIB
- **library** IC_MISC.LIB
- **library** IC_FILE.LIB
- **library** tgl87wc.lib

and

**head file**: ICAMPS11.H

ICAMPS is a commercial software product owned by Ohio Automation Co. If user of QCSMO needs ICAMPS support, please contact Ohio Automation at (614) 592-1810 for detailed information.
LINDO SUBROUTINES are used by QCSMO for solving MLIP models. The following LINDO library functions are required to be linked with QCSMO:

file block.obj
library lindo.lib

The LINDO software is licensed to the Department of Industrial and Systems Engineering, Ohio University for the purpose of research. Any user of QCSMO should contact LINDO Systems Inc. at (312) 871 2524 for permission to link the LINDO library and subroutines.

QCSMO is not copyrighted by the author at this moment. The source code of QCSMO can be tailored for other purposes. The author has no obligation for any problems caused by using the QCSMO source code.