St. Paul University Surigao
St. Paul University System
Surigao City, Philippines

DEVELOPMENT OF A MULTI-DIMENSIONAL PIT OPTIMIZATION SYSTEM FOR STRATEGIC PLANNING IN SURFACE MINES

A Thesis Proposal Presented to

The Faculty of the College of Engineering

St. Paul University Surigao

Surigao City

In Partial Fulfillment

of the Requirements for the Degree

BACHELOR OF SCIENCE IN MINING ENGINEERING

By:

Melba Dominique B. Castorico

November 2023

TABLE OF CONTENTS

CHAPTER 1

THE PROBLEM AND ITS BACKGROUND

INTRODUCTION

Mineral Reserves and Optimized Strategic Plans for Surface Mines

Mine planning software that are utilized in scoping, feasibility, life-of-mine scheduling and in the ongoing re-evaluation of mine plans through production phase. Accurate evaluation of financial viability of the mineral deposit and development, then determination of the optimal long term strategic mine plan and schedule to extract the full value of that deposit over the life of the mine. However, the Algorithm that analysis the conditions for scheduling includes strategic scheduling, detailed cost, price and recovery modeling, multiple scenario analysis, blending, cut-off and simultaneous stockpile optimization and failed to consider the open area for extraction of ore as mandate by existing guidelines. Thus, full potential of mine project is not fully attained. Simultaneous mine production and mine rehabilitation is the major focus on the development of this algorithm for attainment of the optimum life-of-mine.

Optimized Economics

Economic optimization entails striving to acquire the best from economy in terms of profits, production, and utility. In entails maximizing the objective functions which contribute towards the best economic outcome.

Good Governance and Environmental Compliance

Environmental resources is essential in economic advancement and the improvement of society's quality of life, and strong environmental governance in important in achieving goals in sustainable way. Mining assumes governance as key to the environment

and, as a consequence, to the business. The development of the algorithm that will provide the full potential of mine project without compromising the environment is a sustainable attainment of economic growth.

Open Area Restriction

In Surface mining, the maximum area for the extraction of ore at any one time shall be dependent on the scale of the mining operation. For projects that have processing plants, the maximum disturbed area for extraction is one hundred sixty-two (162) hectares or two (2) meridional blocks (DAO 2018-19) **DAO2018-19**. Temporary revegetation or progressive rehabilitation shall be implemented immediately on disturbed area exceeding the maximum disturbed area.

Auditability

Auditability of the source code is significant in order to determine how the algorithm handled the data and how it analysis it and ensure that it is not bias.

CONCEPTUAL FRAMEWORK OF THE STUDY

This study aims develop a life of mine optimizer for surface mines or LOMOPTIM for short.. There will be 3 major milestones in the development cycle of the study as illustrated in Figure 1.1.

The first part is the system design phase which will include use case identification, algorithm design and source code writing. Once the design phase is done, the next stage will be the system testing phase wherein the functionality of LOMOPTIM is as expected and no bugs are present. Should bugs be identified, the algorithm and the source code will be modified. The final part of the study will be the evaluation phase wherein LOMOPTIM will be tested by the strategic mine planners and other technical experts.

STATEMENT OF THE PROBLEM

1. What is the status of the current system for strategic mine planning in terms of
   (a) Processing time in achieving the highest net present value
   (b) Achieving the throughput requirement per product
   (c) Achieving the yearly slices which complies to the environmental requirements stipulated in DAO 2018-19
2. How is the system developed in relation to
   (a) Software stack selection
   (b) Current tools used by the company
   (c) End-user's requirements
3. How does the new system compare to the existing system for strategic mine planning?
4. What is the evaluation of the system by the experts and users in terms of
   (a) Efficiency
   (b) User's experience

SIGNIFICANCE OF THE STUDY

The objective of the study is to produce a working application which will determine an economically optimum yearly mining sequence from an arbitrary start up to the end of the mine life while prioritizing sub-optimal restrictions like a minimum annual product throughput and a maximum open area at the end of every year.

Surface Mine Planners

This study makes it possible to create a repeatable and auditable process in creating life-of-mine plans. The development of a system for a Life of Mine optimizer for surface mines in which mine planners will participate and contribute for the inputs and determination of the optimal condition. This study aims to provides a strategic mining sequence with confidence and a non-bias result that will suffice the needs and expectation for a mining project.

Mining Community

Implementation of optimal system in planning for surface mines will benefit the mining community as this will simplify the compliance on restriction for open areas for surface mine. Maximizing the recovery while prioritizing the implementation of good governance and environmental compliance. Establishment of relationship between environmental stewardship and economic success for the success of a mining project. By minimizing the ecological footprint and enhancing operational efficiency, mining operations can achieve balance that not only benefits the environment but also ensure sustainable and economically viable practices over the long term.

The host Mining Community

This study is in-line with the goal of good governance of the mining industry through the prioritization of legal compliance while also taking in consideration the optimum economics.

Academe

Since the systematic approach of Life of mine optimization will readily available, it will be easier for Academe and other organizations to assess what mining engineering students needs to be taught in order to generate graduates who are qualified and equipped.

SCOPE AND LIMITATION OF THE STUDY

The development of the application focuses only on the case of Taganito Mining Coporation (TMC) which has a partner plant using the High Pressure Acid Leach technology. This plant requires a consistent throughput of limonite ore while TMC's mine product is saprolite ore. Thus, this study will only account for two products for the mine optimization: limonite and saprolite.

In addition, the programming side of the application will be done in Fedora Linux **fedora** since the programming toolkits such as the C++ compilers, Qt libraries, and data processing tools such as Postgresql **postgres** and PostGIS **postgis** are readily available.

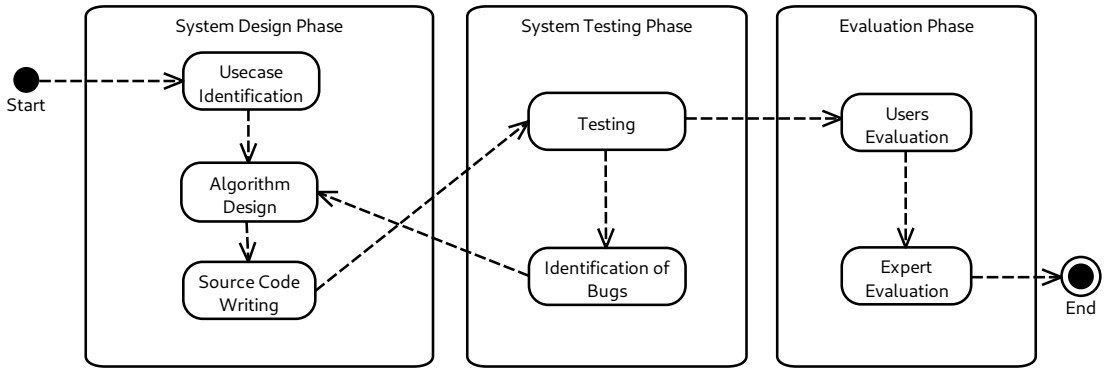

Figure 1.1: Development cycle of LOMOPTIM.

CHAPTER 2

REVIEW OF RELATED LITERATURE

SURFACE MINING

The surface mining method is the most common method in mining due to its cost effectiveness but at the same time, it's visual and environmental impacts are the most conspicuous such as deforestation, dust generation, and surface run-offs. Thus, it is very imperative to minimize to open area in a surface mine in order to also keep the these impacts at its minimum. In this research, one dimension which will be considered during mine optimization is not exceeding the allowable open area at any given time in order to comply with DAO 2018-19 and minimize the environmental impacts.

TAGANITO MINE AND THE LATERITE PROFILE

The mine project which will host this research is Taganito Mine located in Taganito, Claver, Surigao del Norte. Taganito Mine sits on top of a nickeliferous laterite deposit which produces two distinct products: limonite and saprolite. The nickeliferous laterite deposit is produced from the chemical weathring process of ultramafic rocks such as peridotite and dunites. In a mature laterite profile, the layers arranged in the following order from bottom-up: bedrock, saprolite zone, and limonite zone. The bedrock layer is comprised by serpentinized dunites and peridotites with some notable joints and fractures caused by uplifiting. The saprolite zone is where the most nickel is enriched through precipitation during partial dissolution of nickel during weathering. Due to the enrichment of nickel in the saprolite, saprolite ore with sufficient nickel content can be used as ore feed for ferronickel smelting plants. The most chemically altered portion in the laterite profile is the limonite zone. Most magnesium and silica are removed in the limonite zone due to

weathering while the non-mobile elements such as iron, aluminum, chromium are retained as residue. Thus, the limonite zone is usually characterized by high iron and chromium content. Because of this, the ore from limonite can be used as ore feed for iron smelting plants and a source for chromite. In the case of Taganito Mine, the limonite is exclusively sold to a High Pressure Acid Leaching Plant as an ore feed for the production of Ni-Co mixed sulfide.

EXISTING TOOLS AND ALGORITMS

Lerch-Grossman

The most common algoritm used in mine optimization is the algorithm described by Helmut Lerch and Ingo Grossman. This can be illustrated with each mine block with both downward force (cost) and upward force (revenue value). The objective of this algorithm is to maximize the total upward force without accounting for the mining sequence **IMS**.

Pseudoflow

A variation of the Lerch-Grossman algorithm (LG) which adapted a network analysis of directed graphs and spanning trees is called the pseudoflow. Although there are also lots of variations of this algoritm, the best variants chosen in most literatures are lowest label and highest label methods **pseudoflow**.

GEOVIA Whittle

GEOVIA Whittle is proprietary strategic mine planning tool which is commonly used in the Phillipines by various surface mines. This application implements the above-mentioned algorithms in mine optimization, Lerch-Grossman and Pseudoflow, while maximizing the Net Present Value of the generated annual pit shells **whittle**.

Limitations of GEOVIA Whittle and Its Algorithms

Although commonly used in the Phillipines, GEOVIA Whittle at present cannot conform to the recent developments of laws related to mining such as the DAO 2018-19 **DAO2018-19** which limits the open area of all mines since GEOVIA Whittle focuses only on addressing engineering design constraints and economic optimization. In addition, GEOVIA Whittle is able to consider a minimum extraction limit but cannot set a minimum production limit for a specific product. In the present mineral prices, saprolite ore is more valuable than limonite ore so GEOVIA Whittle will prioritize saprolite ore in the annual production instead of the limonite ore. However, in the case of Taganito Mine where a minimum production of limonite ore is required to sustain a High Pressure Acid Leach plant, using GEOVIA Whittle will only a produce an unacceptable yearly production.

DEVELOPMENT TOOLS

C++

The programming language to be used in the development of LOMOPTIM will be C++. This a high level language which applies object-oriented programming for easier data abstraction. It also implements easier memory management through the use smart pointers and efficient use of multi-threaded machines through threading and mutual exclusions **cpp**.

Qt

In the creation of the graphical user interface of LOMOPTIM, the C++ toolkit called Qt will be used. Qt is a graphical toolkit built for C++ which already contains a lot of libraries of graphical elements. Included in this toolkit is the graphical designer which

will allow the reaserchers to design an interface without writing a single line of code **qt**.

CHAPTER 3

METHODS

RESEARCH DESIGN

This study aims to design and develop an application which will mainly comprise of systems designing and writing of software components. During the initial part of the research, the descriptive-developmental design will be applied in order to understand the existing system used in strategic mine planning and identify the gaps. During the next phase of the research which is the system design phase,, a multi-phase iterative approach as discussed by Leedy and Ormrod will be applied. This research design is used commonly in design-based research wherein the researchers apply existing knowledge to create more effective systems **MixedMethod**.

RESPONDENTS

LOMOPTIM will be tested mostly by the mine planners in Taganito Mine and possibly other mine planners of other operating mines of Nickel Asia Corporation. The researchers will formulate a method of preparing the sample data input. The output of LOPTIM will then be compared with in GEOVIA Whittle by the participating mine planners.

METHODOLOGY

System Design

Use Case

The basic use case **usecase** of LOMOPTIM is illustrated in Figure 3.1. The user will input both the block model file and various engineering parameters which will be enumerated further during the study. Some identified parameters are bench width, bench

height, bench slope, extraction limit in wet metric tons, maximum open areas in hectares, and minimum extt tonnage After completion of inputs, the user can run the algorithm which will output the yearly pit shells and the yearly production details which should include tonnages and grades.

File Structures

The block model input will be composed of 3 comma-delimited files (CSV) which just contain ASCII texts. The first CSV file will be comprised by the collar data of the block model which will contain the 3D coordinates. The second CSV file will be comprised by the block model itself which will contain the chemical characteristics of block down each collar with its quantified extraction cost and revenue values. The extraction cost and revenue values will be prepared beforehand while each block may have different rock types at varying compenents. The last CSV file will contain the rock type or product type along with their corresponding bulk densities and moisture contents.

Algorithm

The algorithm to be used in LOMOPTIM will be a brute-force approach wherein each mining sequence will be explored if it conforms to the engineering constraint inputted by the user. All iteration will be predetermined using permutation included in the C++ standard library through parallel computation (threading). Each finished iteration which conforms to the engineering constraints and also results to the a higher net present value will be saved through mutual exclusion to prevent race conditions of each iteration.

Source Code Writing

The source code of LOMOPTIM will be solely written in C++ **cpp** in the development environment called Qt Creator which is part of the Qt toolkit **qt**.

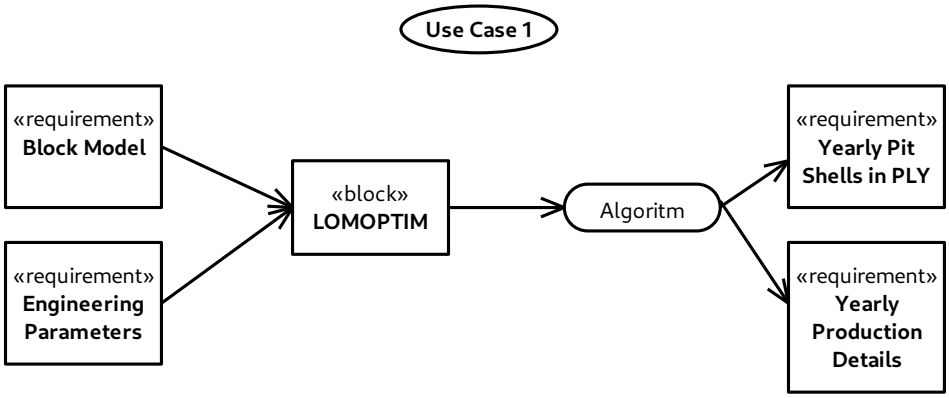

Figure 3.1: Initial use case of LOMOPTIM.

INSTRUMENTS

During the development of LOMOPTIM, free and open source tools will be used as discussed in the review of related literature.

In the method of qualitative evaluation, a similar method based on the ER Mine Tracer research design will be applied wherein the researchers of the said study used survey to qualitatively evaluate ER Mine Tracer's functionality and efficiency **ERMineTracer**.

In the preparation of data, data management tools such as PostreSQL **postgres**, PostGIS **postgis**, and R **R** will be used.

DATA GATHERING PROCEDURE

Quantitative Data

The efficiency and effectiveness of LOMOPTIM will be evaluated in terms of compliance with the input restrictions and processing time.

Qualitative Data

Qualitative data will be collected through the use of questionnaires to the potential users, engineers, and technology experts if available. Each question will be have a corresponding point system similar to the Liker 4-point scale **Likert**.

DATA ANALYSIS

The performance of LOMOPTIM will be compared to Whittle which is the existing system used by TMC. The researchers will be using R version 4.3 **R** to analyze the quantitative data.

Questionnaire surveys with qualitative information will be distributed to Strategic Mine Planners, IT Experts, Computer Engineers and other experts on this topic in order to

collect data that are qualitative in nature.

The result of the surveys and the collected quantitative data will be used by the researchers to determine if LOMOPTIM meets the user's requirements, meets the standard's of the technical experts, and does not have any technical limitations that might hinder its adoption.

CHAPTER 4

RESULTS AND DISCUSSION

Surface Mines has a lot of mandated laws and guidelines to follow that limits the full potential of current Mine Planning Software in generating Optimal Life of Mine plans. This leads to the development of an algorithm that would suffice the requirements to attain the most profitable and sustainable Mining Project. General workflow of the algorithm is shown in Figure 4.1.

INPUT DATA

To run the algorithm, the user must specify the data requirements: Block model and Engineering Parameters. The Block model is the collection of several mining area, each contains specific commodity content and corresponding revenue. It also contain the geological characteristic of the mining project. In which the objective of the development of this algorithm is to determine the most profitable sequence of mining the area, considering the various predetermined parameters. By optimizing the revenue potential and specific constraints, the algorithm aims to maximize the profitability of the mining project. This is designed for efficient resource allocation and enhance revenue generation, and improve the overall operational effectiveness in resource extraction.

DATA VALIDATION

The algorithm needs to verify that the input block model data contains the necessary attributes and the predefined parameters are within the acceptable ranges. Otherwise, the algorithm prompts the user to rectify the error by entering a valid data. This is done to eliminate errors in generating the Optimal Mining Sequence.

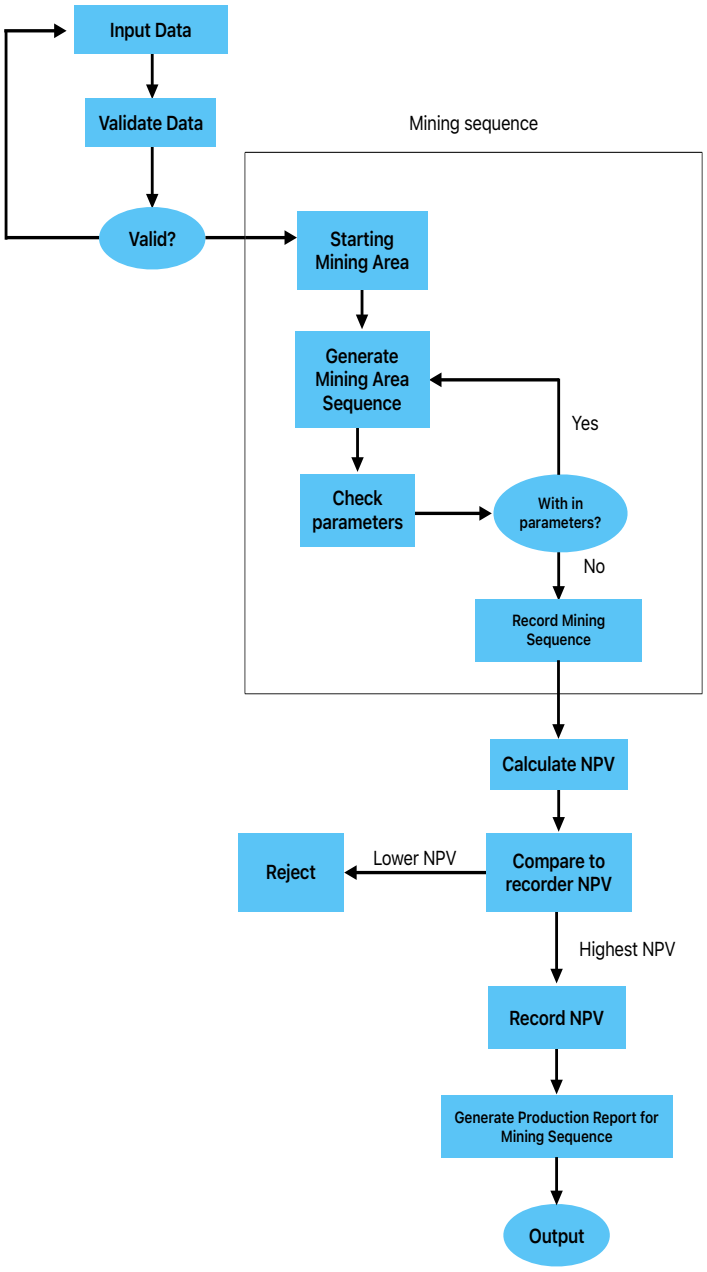

Figure 4.1: High level flow of LOMOPTIM algorithm.

GENERATION OF MINING SEQUENCE

Multiple Mining sequence is generated for the purpose of determining all the possible outcome of a mining project. The algorithm continues to generate the sequence until it encounters the point where the sequence is no longer valid according to the predetermined parameters. This iterative process will continue until all possible mining sequence has been considered.

OPTIMAL NET PRESENT VALUE

Each Mining Sequence generated in the algorithm has a corresponding production details such as the tonnages, commodities and revenues. These details can be used to calculate the Net Present Value (NPV), a fundamental measure used in assessing the profitability of a Mining Project. NPV computation involves discounting the expected future revenues of the mining sequence to their present value given to a certain discount rate. This methods enables the comparison of revenues generated over the projects duration in present-day term. NPV is computed by:

$$NPV = \sum_{t=0}^{n} \frac{Rt}{(1+i)^t}$$

[h] where :

Rt : Net cash inflow/ revenue during a single period, t

i : Discount rate

T : Number of timer periods

NPV Calculation for Individual Mining Area

Every resulting Mining Sequence is comprise with multiple number of Mining Area. The calculation of NPV is based one the revenue by series of mining area.

Discounting Cash Flows to Present Value

Once the expected revenue for each mining area are determine, the algorithm discounts these future revenue to their present value using the NPV method. The discounting process adjusts the value of future revenue to reflect their worth the the present day term, accounting for the value of money and the project's specific discount rate.

Summation of NPV

After computing the NPV for each mining area within the mining sequence, the algorithm sumps up these NPVs to obtain the total NPV for the entire mining sequence. By summing up the NPVs across the mining areas, the algorithm evaluates the profitability of the mining sequence, considering the combined revenue potential.

SELECTION OF THE OPTIMAL MINING SEQUENCE

The summation of NPVs per Mining Sequence is noted and compared to the recorded NPV. The mining sequence with the highest total NPV represents the most profitable and sustainable life of mine plans among the options considered. The Optimal Mining Sequence will be extracted in form of production report and in yearly pit shell.

Appendices

SOURCE CODE

BLOCKMODEL.CPP

```
#include "blockmodel.h"
#include <QDateTime>
#include <QFile>
#include <QMessageBox>

MiningArea::MiningArea(const double& x_coordinates,
                       const double& y_coordinates,
                       const double& z_coordinates,
                       const size_t& id,
                       QTextBrowser *log)
{
    this->id = id;
    this->log = log;
    this->x = x_coordinates;
    this->y = y_coordinates;
    this->z = z_coordinates;
}

MiningArea::~MiningArea(){}

box_2d MiningArea::asBox2D(const double& size_x, const double& size_y) const
{
    box_2d area(point_2d(this->x - size_x - (size_x / 2), this->y - size_y - (size_y / 2)
        ),
                point_2d(this->x + size_x + (size_x / 2), this->y + size_y + (size_y / 2)
                    ));
    return area;
}

point_value MiningArea::asPointValue() const
{
    return std::make_pair(this->asPoint2D(), this->getId());
```

```
}

point_2d MiningArea::asPoint2D() const
{
    return point_2d(this->x, this->y);
}

size_t MiningArea::blockCount() const
{
    return this->blocks.size();
}

size_t MiningArea::getId() const
{
    return this->id;
}

void MiningArea::appendLog(const QString& message)
{
    this->log->append(QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss ") +
        message);
}

void MiningArea::setAdjacentArea(std::vector<std::shared_ptr<MiningArea>> adjacent_areas)
{
    this->adjacent_areas = adjacent_areas;
}

BlockModel::BlockModel(const QString& blocks,
                       const QString& centroids,
                       const QString& rocks,
                       const double& x,
                       const double& y,
                       const double& z,
```

```
                        QTextBrowser *log)
{
    this->log = log;
    this->size_x = x;
    this->size_y = y;
    this->size_z = z;

    QString guide;
    QString line;
    QStringList csv_entry;

    unsigned char valid_centroid = 4;

    QFile centroid_csv(centroids);
    if (!centroid_csv.open(QIODevice::ReadOnly | QIODevice::Text))
    {
        this->appendLog("Error: Could not open centroid file.");
    }
    else
    {
        QTextStream in_centroid(&centroid_csv);
        line = in_centroid.readLine();
        csv_entry = line.split(",");
        guide = "Acceptable Collar CSV should only contain the following headers in order
            :\nid, x, y, z";
        if (csv_entry.size() != 4)
        {
            QMessageBox::information(NULL, "Message", guide);
        }
        else
        {
            if (csv_entry.at(0) != QString("id"))
            {
                guide += QString("\n1st header should be 'id'.");
```

```
        QMessageBox::information(NULL, "Message", guide);
    }
    else
    {
        valid_centroid--;
    }
    if (csv_entry.at(1) != QString("x"))
    {
        QMessageBox::information(NULL, "Message", QString("2nd header shoud be 'x
            '."));
    }
    else
    {
        valid_centroid--;
    }
    if (csv_entry.at(2) != QString("y"))
    {
        QMessageBox::information(NULL, "Message", QString("2nd header shoud be 'y
            '."));
    }
    else
    {
        valid_centroid--;
    }
    if (csv_entry.at(3) != QString("z"))
    {
        QMessageBox::information(NULL, "Message", QString("2nd header shoud be 'z
            '."));
    }
    else
    {
        valid_centroid--;
    }
    if (valid_centroid == 0)
```

```
{
    long idx = -1;
    while (!in_centroid.atEnd())
    {
        line = in_centroid.readLine();
        csv_entry = line.split(",");
        if (csv_entry.at(0).toLong() > (idx + 1))
        {
            if (csv_entry.at(0).toLong() - (idx + 1) > 1)
            {
                this->appendLog("Error: id column skipped at "
                                + csv_entry.at(0)
                                + ".");
                valid_centroid++;
                break;
            }
            idx = csv_entry.at(0).toLong() - 1;
            this->mining_areas.push_back(std::make_shared<MiningArea>(
                csv_entry.at(1).toDouble(),
                                                    csv_entry
                                                        .at
                                                        (2)
                                                        .
                                                        toDouble
                                                        (),
                                                    csv_entry
                                                        .at
                                                        (3)
                                                        .
                                                        toDouble
                                                        (),
                                                    (size_t
                                                        )
                                                        idx
```

```
                                                                    ,
                                                                log));
                    this->insertIndex(this->mining_areas[(size_t) idx]->asPointValue
                        ());
                }
                else if ((csv_entry.at(0).toLong() == (idx + 1)))
                {
                    this->appendLog("Error: id column duplicated at "
                                    + csv_entry.at(0)
                                    + ".");
                    valid_centroid++;
                    break;
                }
                else
                {
                    this->appendLog("Error: id column in collar csv is not sorted in
                        ascending order.");
                    valid_centroid++;
                    break;
                }
            }
        }
    }
}
centroid_csv.close();

unsigned char valid_rock = 4;

QFile rock_csv(rocks);
if (!rock_csv.open(QIODevice::ReadOnly | QIODevice::Text))
{
    this->appendLog("Error: Could not open rock information file.");
}
else
```

```
{
    QTextStream in_rock(&rock_csv);
    line = in_rock.readLine();
    csv_entry = line.split(",");
    guide = "Acceptable Rock Info CSV should only contain the following headers in
        order:\nid, name, dry_density, moisture";
    if (csv_entry.size() != 4)
    {
        QMessageBox::information(NULL, "Message", guide);
    }
    else
    {
        if (csv_entry.at(0) != QString("id"))
        {
            guide += QString("\n1st header should be 'id'.");
            QMessageBox::information(NULL, "Message", guide);
        }
        else
        {
            valid_rock--;
        }
        if (csv_entry.at(1) != QString("name"))
        {
            QMessageBox::information(NULL, "Message", QString("2nd header shoud be '
                name'."));
        }
        else
        {
            valid_rock--;
        }
        if (csv_entry.at(2) != QString("dry_density"))
        {
            QMessageBox::information(NULL, "Message", QString("2nd header shoud be '
                dry_density'."));
```

```
}
else
{
    valid_centroid--;
}
if (csv_entry.at(3) != QString("z"))
{
    QMessageBox::information(NULL, "Message", QString("2nd header shoud be 'z
        '"));
}
else
{
    valid_centroid--;
}
if (valid_centroid == 0)
{
    long idx = -1;
    while (!in_centroid.atEnd())
    {
        line = in_centroid.readLine();
        csv_entry = line.split(",");
        if (csv_entry.at(0).toLong() > (idx + 1))
        {
            if (csv_entry.at(0).toLong() - (idx + 1) > 1)
            {
                this->appendLog("Error: id column skipped at "
                                + csv_entry.at(0)
                                + ".");
                valid_centroid++;
                break;
            }
            idx = csv_entry.at(0).toLong() - 1;
            this->mining_areas.push_back(std::make_shared<MiningArea>(
                csv_entry.at(1).toDouble(),
```

```
                                                                csv_entry
                                                                    .at
                                                                    (2)
                                                                    .
                                                                    toDouble
                                                                    (),
                                                                csv_entry
                                                                    .at
                                                                    (3)
                                                                    .
                                                                    toDouble
                                                                    (),
                                                                (size_t
                                                                    )
                                                                    idx
                                                                    ,
                                                                log));
        this->insertIndex(this->mining_areas[(size_t) idx]->asPointValue
            ());
    }
    else if ((csv_entry.at(0).toLong() == (idx + 1)))
    {
        this->appendLog("Error: id column duplicated at "
                        + csv_entry.at(0)
                        + ".");
        valid_centroid++;
        break;
    }
    else
    {
        this->appendLog("Error: id column in collar csv is not sorted in
            ascending order.");
        valid_centroid++;
        break;
```

```
                    }
                }
            }
        }
    }
    centroid_csv.close();

    if (valid_centroid == 0)
    {
        this->populateAdjacentMiningArea();
        this->initialized = 1;
    }
}

BlockModel::~BlockModel(){}

bool BlockModel::isInitialized() const
{
    return this->initialized;
}

size_t BlockModel::areaCount() const
{
    return this->mining_areas.size();
}

size_t BlockModel::blockCount() const
{
    size_t block_count = 0;
    for (auto& mining_area : this->mining_areas)
    {
        block_count += mining_area->blockCount();
    }
    return block_count;
```

```
}

std::vector<std::shared_ptr<MiningArea>> BlockModel::getAdjacentAreas(std::shared_ptr<
    MiningArea> mining_area)
{
    std::vector<std::shared_ptr<MiningArea>> area_list;
    std::vector<point_value> adjacent_values;
    this->index.query(boost::geometry::index::intersects(mining_area->asBox2D(this->
        size_x, this->size_y)), std::back_inserter(adjacent_values));
    for (auto& adjacent_value : adjacent_values)
    {
        if (mining_area->getId() != std::get<1>(adjacent_value))
        {
            area_list.push_back(this->mining_areas[std::get<1>(adjacent_value)]);
        }
    }
    return area_list;
}

void BlockModel::appendLog(const QString& message)
{
    this->log->append(QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss ") +
        message);
}

void BlockModel::insertIndex(const point_value& pv)
{
    this->index.insert(pv);
}

void BlockModel::populateAdjacentMiningArea()
{
    for (auto& mining_area : this->mining_areas)
    {
```

```
            mining_area->setAdjacentArea(this->getAdjacentAreas(mining_area));
        }
    }
```

BLOCKMODEL.H

```
#ifndef BLOCKMODEL_H
#define BLOCKMODEL_H

#include "boost.h"
#include <QString>
#include <QTextBrowser>
#include <vector>

class Block
{
    // volume fractions per rock type component
    std::vector<double> component_fractions;

    // Cost in PhP per WMT if mined as product
    std::vector<double> mining_costs;

    // Cost in PhP per WMT if mined as waste
    std::vector<double> waste_costs;

    // Revenue in PhP per WMT
    std::vector<double> revenues;

    // Elemental constituents per rock type.
    // The primary dimension of the vector defines the rock type while the secondary
        dimension is the elemental constituent in percentage.
    std::vector<std::vector<float>> elemental_consituents;

public:
    Block();
```

```
    ~Block();

    double profit() const;
    std::vector<double> wmt() const;
};


class MiningArea
{
    QTextBrowser *log;

    // Centroid coordinates of the mining area.
    double x;
    double y;

    // elevation of the centroid of the top block
    double z;

    std::vector<std::shared_ptr<Block>> blocks;
    std::vector<std::shared_ptr<MiningArea>> adjacent_areas;
    size_t id;

public:
    MiningArea(const double& x_coordinates,
               const double& y_coordinates,
               const double& z_coordinates,
               const size_t& id,
               QTextBrowser *log);
    ~MiningArea();

    // Returns the bounding box of the area which also covers nearby mining areas.
    // The dimensions of the bounding box will be X = 2 * size_x and Y = 2 * size_y.
    // The centroid of the bounding is the centroid of the mining area.
    box_2d asBox2D(const double& size_x, const double& size_y) const;
```

```
    // Returns the centroid of the mining area.
    point_2d asPoint2D() const;

    // Returns the centroid along with the id of the mining area.
    point_value asPointValue() const;

    // Returns the number of blocks within the mining area.
    size_t blockCount() const;

    // Returns the id of the mining area.
    size_t getId() const;

    // Returns a vector with length blockCount() containing identical getId()
    std::vector<size_t> getVectorizedId();

    // Logs a message with time stamp.
    void appendLog(const QString& message);

    // Sets adjacent_areas;
    void setAdjacentArea(std::vector<std::shared_ptr<MiningArea>> adjacent_areas);
};


class SimulatedMiningArea
{
    std::shared_ptr<MiningArea> mining_area;
    unsigned char excavated;

public:
    SimulatedMiningArea(std::shared_ptr<MiningArea> mining_area);
    ~SimulatedMiningArea();

    // Checks whether the exposed area can be excavated by applying the design parameters
```

```
        constraints.
    bool canBeExcavated() const;
};


class BlockModel
{
    QTextBrowser *log;

    bool initialized = 0;

    // Maximum allowable height of bench
    double bench_height;

    // Angle of the bench
    double bench_slope;

    // Annual mining limit in wet metric tons
    double mining_limit;

    // Block dimensions in meters.
    double size_x;
    double size_y;
    double size_z;

    // An R-tree index to be used for querying adjacent areas.
    rtree_t index;

    std::vector<QString> rock_types;
    std::vector<QString> elemental_contituents;

    // Container for all mining areas.
    std::vector<std::shared_ptr<MiningArea>> mining_areas;
```

```
    // Densities per rock type in tons per cubic meter.
    std::vector<double> dry_densities;

    // Required annual mining production per rock type in wet metric tons.
    std::vector<double> minimum_production;

    // Assumed insity moisture content of each rock type in percentage.
    std::vector<double> moisture_content;

public:
    BlockModel(const QString& blocks,
               const QString& centroids,
               const QString& rocks,
               const double& x,
               const double& y,
               const double& z,
               QTextBrowser *log);
    ~BlockModel();

    // Checks the BlockModel if properly initialized.
    bool isInitialized() const;

    // Returns the number of areas in the BlockModel.
    size_t areaCount() const;

    // Returns the number of blocks in the BlockModel.
    size_t blockCount() const;

    // Returns the vector of MiningAreas adjacent to the input MiningArea.
    std::vector<std::shared_ptr<MiningArea>> getAdjacentAreas(std::shared_ptr<MiningArea>
        mining_area);

    // Logs a message with time stamp.
    void appendLog(const QString& message);
```

```
    // Inserts a point_value to the R-Tree index.
    void insertIndex(const point_value& pv);

    // Builds the vector of adjacent areas per mining area.
    void populateAdjacentMiningArea();
};

#endif // BLOCKMODEL_H
```

BOOST.H

```
#ifndef BOOST_H
#define BOOST_H

#include <boost/geometry.hpp>
#include <boost/geometry/index/rtree.hpp>

typedef boost::geometry::model::point<double, 2, boost::geometry::cs::cartesian> point_2d
    ;
typedef std::pair<point_2d, size_t> point_value;
typedef boost::geometry::model::box<point_2d> box_2d;
typedef boost::geometry::index::rtree<point_value, boost::geometry::index::quadratic<16>>
     rtree_t;

#endif // BOOST_H
```

MAIN.CPP

```
#include "mainwindow.h"

#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
```

```
    MainWindow w;
    w.show();
    return a.exec();
}
```

MAINWINDOW.CPP

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QDateTime>
#include <QFileDialog>
#include <QMessageBox>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->sizeX->setValue(10);
    ui->sizeY->setValue(10);
    ui->sizeZ->setValue(3);
    ui->log->show();
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::appendLog(const QString& message)
{
    ui->log->append(QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss ") +
        message);
}
```

```
void MainWindow::on_clearButton_clicked()
{
    ui->log->clear();
}

void MainWindow::on_pushButtonBlockData_clicked()
{
    QString filePath = QFileDialog::getOpenFileName(this, "Open File", QString(), "CSV
        Files (*.csv)");
    ui->fileBlock->setText(filePath);
}

void MainWindow::on_pushButtonCollarCentroids_clicked()
{
    QString filePath = QFileDialog::getOpenFileName(this, "Open File", QString(), "CSV
        Files (*.csv)");
    ui->fileCollar->setText(filePath);
}

void MainWindow::on_pushButtonRockInfo_clicked()
{
    QString filePath = QFileDialog::getOpenFileName(this, "Open File", QString(), "CSV
        Files (*.csv)");
    ui->fileRock->setText(filePath);
}

void MainWindow::on_pushButtonInitialize_clicked()
{
    if (this->ui->fileBlock->text().isEmpty())
    {
        QMessageBox::information(this, "Message", QString("No selected block model file."
            ));
        return;
    }
```

```
if (this->ui->fileCollar->text().isEmpty())
{
    QMessageBox::information(this, "Message", QString("No selected collar data file."
        ));
    return;
}
if (this->ui->fileRock->text().isEmpty())
{
    QMessageBox::information(this, "Message", QString("No selected rock data file."))
        ;
    return;
}
if (this->ui->sizeX->text().toDouble() < 0.01)
{
    QMessageBox::information(this, "Message", QString("Size along X-axis should be
        greater than 0.00."));
    return;
}
if (this->ui->sizeY->text().toDouble() < 0.01)
{
    QMessageBox::information(this, "Message", QString("Size along Y-axis should be
        greater than 0.00."));
    return;
}
if (this->ui->sizeZ->text().toDouble() < 0.01)
{
    QMessageBox::information(this, "Message", QString("Size along Z-axis should be
        greater than 0.00."));
    return;
}
this->blockmodel = std::make_shared<BlockModel>(this->ui->fileBlock->text(),
                                                this->ui->fileCollar->text(),
                                                this->ui->fileRock->text(),
                                                this->ui->sizeX->text().toDouble(),
```

```
                                                    this->ui->sizeY->text().toDouble(),
                                                    this->ui->sizeZ->text().toDouble(),
                                                    this->ui->log);
    if (this->blockmodel->isInitialized())
    {
        this->appendLog("Block Model initialized.");
        if (this->blockmodel->areaCount() > 0)
        {
            this->appendLog(QString("Area Count: ")
                            + QString::number(this->blockmodel->areaCount())
                            + QString(". Block Count: ")
                            + QString::number(this->blockmodel->blockCount()));
        }
    }
    else
    {
        this->appendLog("Block Model failed to initialized.");
    }
}
```

MAINWINDOW.H

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include "blockmodel.h"
#include <QMainWindow>
#include <memory>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
```

```
    Q_OBJECT

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    void appendLog(const QString& message);

private slots:
    void on_clearButton_clicked();
    void on_pushButtonCollarCentroids_clicked();
    void on_pushButtonInitialize_clicked();
    void on_pushButtonBlockData_clicked();
    void on_pushButtonRockInfo_clicked();

private:
    Ui::MainWindow *ui;
    std::shared_ptr<BlockModel> blockmodel;
};
#endif // MAINWINDOW_H
```

MAINWINDOW.UI

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>628</width>
    <height>487</height>
   </rect>
  </property>
  <property name="windowTitle">
```

```
 <string>Life of Mine Optimizer</string>
</property>
<widget class="QWidget" name="centralwidget">
 <layout class="QGridLayout" name="gridLayout_5">
  <item row="0" column="0">
   <widget class="QTabWidget" name="tabWidget">
    <property name="currentIndex">
     <number>0</number>
    </property>
    <widget class="QWidget" name="tab">
     <attribute name="title">
      <string>Block Model Data</string>
     </attribute>
     <layout class="QGridLayout" name="gridLayout_4">
      <item row="1" column="0">
       <widget class="QPushButton" name="pushButtonInitialize">
        <property name="minimumSize">
         <size>
          <width>400</width>
          <height>0</height>
         </size>
        </property>
        <property name="font">
         <font>
          <bold>true</bold>
         </font>
        </property>
        <property name="text">
         <string>Initialize</string>
        </property>
       </widget>
      </item>
      <item row="0" column="0">
       <widget class="QWidget" name="widget" native="true">
```

```
<layout class="QGridLayout" name="gridLayout_3">
 <item row="0" column="0">
  <widget class="QGroupBox" name="groupBox_2">
   <property name="title">
    <string>File Import</string>
   </property>
   <layout class="QGridLayout" name="gridLayout_2">
    <item row="0" column="1">
     <widget class="QPushButton" name="pushButtonCollarCentroids">
      <property name="autoFillBackground">
       <bool>false</bool>
      </property>
      <property name="text">
       <string>Select Collar Centroids CSV</string>
      </property>
     </widget>
    </item>
    <item row="1" column="1">
     <widget class="QPushButton" name="pushButtonBlockData">
      <property name="text">
       <string>Select Block Data CSV</string>
      </property>
     </widget>
    </item>
    <item row="2" column="0">
     <widget class="QLineEdit" name="fileRock"/>
    </item>
    <item row="2" column="1">
     <widget class="QPushButton" name="pushButtonRockInfo">
      <property name="text">
       <string>Select Rock Info CSV</string>
      </property>
     </widget>
    </item>
```

```
    <item row="0" column="0">
     <widget class="QLineEdit" name="fileCollar"/>
    </item>
    <item row="1" column="0">
     <widget class="QLineEdit" name="fileBlock"/>
    </item>
   </layout>
  </widget>
 </item>
 <item row="0" column="1">
  <widget class="QGroupBox" name="groupBox">
   <property name="maximumSize">
    <size>
     <width>100</width>
     <height>16777215</height>
    </size>
   </property>
   <property name="title">
    <string>Block Size (m)</string>
   </property>
   <layout class="QGridLayout" name="gridLayout">
    <item row="2" column="0">
     <widget class="QDoubleSpinBox" name="sizeZ"/>
    </item>
    <item row="1" column="0">
     <widget class="QDoubleSpinBox" name="sizeY"/>
    </item>
    <item row="2" column="1">
     <widget class="QLabel" name="label_3">
      <property name="text">
       <string>Z</string>
      </property>
     </widget>
    </item>
```

```
<item row="0" column="0">
 <widget class="QDoubleSpinBox" name="sizeX"/>
</item>
<item row="0" column="1">
 <widget class="QLabel" name="label">
  <property name="text">
   <string>X</string>
  </property>
 </widget>
</item>
<item row="1" column="1">
 <widget class="QLabel" name="label_2">
  <property name="text">
   <string>Y</string>
  </property>
 </widget>
</item>
</layout>
</widget>
</item>
</layout>
</widget>
</item>
</layout>
</widget>
<widget class="QWidget" name="tab_2">
 <attribute name="title">
  <string>Design Parameters</string>
 </attribute>
</widget>
<widget class="QWidget" name="tab_4">
 <attribute name="title">
  <string>Logs</string>
 </attribute>
```

```
<layout class="QVBoxLayout" name="verticalLayout">
 <item>
  <widget class="QTextBrowser" name="log"/>
 </item>
 <item>
  <widget class="QPushButton" name="clearButton">
   <property name="text">
    <string>Clear Logs</string>
   </property>
  </widget>
 </item>
</layout>
</widget>
<widget class="QWidget" name="tab_3">
 <attribute name="title">
  <string>Help</string>
 </attribute>
 <layout class="QGridLayout" name="gridLayout_6">
  <item row="0" column="0">
   <widget class="QTextBrowser" name="textBrowser">
    <property name="html">
     <string>&lt;!DOCTYPE HTML PUBLIC &quot;-//W3C//DTD HTML 4.0//EN&quot; &quot;
        http://www.w3.org/TR/REC-html40/strict.dtd&quot;&gt;
&lt;html&gt;&lt;head&gt;&lt;meta name=&quot;qrichtext&quot; content=&quot;1&quot; /&gt;&
    lt;meta charset=&quot;utf-8&quot; /&gt;&lt;style type=&quot;text/css&quot;&gt;
p, li { white-space: pre-wrap; }
hr { height: 1px; border-width: 0; }
li.unchecked::marker { content: &quot;\2610&quot;; }
li.checked::marker { content: &quot;\2612&quot;; }
&lt;/style&gt;&lt;/head&gt;&lt;body style=&quot; font-family:'Cantarell'; font-size:11pt;
     font-weight:400; font-style:normal;&quot;&gt;
&lt;p style=&quot; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px;
    -qt-block-indent:0; text-indent:0px;&quot;&gt;&lt;span style=&quot; font-size:16pt;
    font-weight:700;&quot;&gt;Life of Mine Optimizer (LOMOPTIM)&lt;/span&gt;&lt;/p&gt;
```

<p style="-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:14pt;">
</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">LOMOPTIM is a simple strategic mine planning tool in determining the optimium economics of an open pit mine. It uses 3 input csv's containing the block model data. The design parameters are also inputed which are the basis for the constaints of the mining sequence. It outputs the yearly pit shells in PLY format along with a summary table of the annual production.</p>
<p style="-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">
</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">CSV Inputs</p>
<p style="-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px; font-size:14pt;">
</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">Collar CSV</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">- Only columns id, x, y, z are accepted </p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">- Column id must be in ascending order</p>
<p style=" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;">- Column id must start with the value 1</p></body></html></string>
</property>
</widget>
</item>

```
        </layout>
       </widget>
      </widget>
     </item>
    </layout>
   </widget>
   <widget class="QMenuBar" name="menubar">
    <property name="geometry">
     <rect>
      <x>0</x>
      <y>0</y>
      <width>628</width>
      <height>23</height>
     </rect>
    </property>
   </widget>
   <widget class="QStatusBar" name="statusbar"/>
  </widget>
  <resources/>
  <connections/>
</ui>
```

CURRICULUM VITAE

MELBA DOMINIQUE B. CASTORICO
Bonbon, Butuan City, Agusan del Norte
melba.dominque@gmail.com

PERSONAL INFORMATION

DATE OF BIRTH : June 1, 1992
PLACE OF BIRTH : Butuan City
AGE : 31
GENDER : Female
NATIONALITY : Filipino
RELIGION : Catholic
CIVIL STATUS : Married
FATHER'S NAME : Michael B. Burdeos
MOTHER'S NAME : Josephine A. Burdeos

EDUCATIONAL BACKGROUND

COLLEGE : Saint Paul University Surigao
: Corner Rizal and San Nicolas Streets, Surigao City
HIGH SCHOOL : Agusan National High School
: A.D. Curato St., Butuan City
ELEMENTARY : La Trinidad Elementary School
: Bonbon, Butuan City