

# **OPTIMERA CONSTRAINT OPTIMIZATION SOLVER**

**By**

**Atiqa Zafar**

**NUST201200024BSEECSS60512F**

**Zaryab Khan**

**NUST201201122BSEECSS60512F**



**A PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT  
OF THE REQUIREMENT FOR THE DEGREE OF  
BACHELORS OF ENGINEERING IN SOFTWARE ENGINEERING**

**Department of Computing**

**SCHOOL OF ELECTRICAL ENGINEERING & COMPUTER  
SCIENCE  
NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY  
ISLAMABAD, PAKISTAN  
2016**

## CERTIFICATE

It is certified that the contents and form of thesis entitled “**OPTIMERA: Constraint Optimization Solver**” submitted by *Atiqa Zafar (NUST 2012 00024 BSECS 60512F)* and *Zaryab Khan (NUST 2012 01122 BSECS 60512F)* have been found satisfactory for the requirement of the degree.

**Advisor:** \_\_\_\_\_

**(Dr. Muhammad Moazam Fraz)**

**Co-Advisor:** \_\_\_\_\_

**(Dr. Mian Muhammad Hamayun)**

## **DEDICATION**

To Allah the Almighty

&

To our Parents and Faculty

## **ACKNOWLEDGEMENTS**

We are extremely thankful to our Advisor Dr. Moazam Fraz and Co- Advisor Dr. Mian Hamayun for supporting us throughout the development of this project. This project would not have been possible without their guidance and reassurance.

We are also thankful to Sir Ghulam Hassan Shami and Sir Muhammad Waqas who provided us with important data and counseling that helped us greatly in developing our project. I would further like to acknowledge the efforts of Dr. Aakash Ahmad Abbasi and Madam Hirra Anwar who examined our project during the Design Expo.

# TABLE OF CONTENTS

ABSTRACT .....	1
INTRODUCTION .....	2
1.1. Document Purpose .....	2
1.2. Intended Audience and Document Overview .....	3
1.3. Problem Statement and Motivation for the New System .....	5
1.4. Product Overview .....	6
LITERATURE REVIEW .....	8
2.1. Key Concepts .....	8
2.1.1. Constraint Optimization .....	8
2.1.2. Modelling with Linear Programming .....	9
2.1.3. How do solvers work? .....	13
2.2. Available Market Solutions .....	14
2.3. Technology Review .....	19
2.3.1. Open Source Solvers .....	19
2.3.2. Commercial LP Solvers .....	21
2.3.3 Performance Comparison .....	21
FUNCTIONALITY AND DESIGN .....	24
3.1 Product Functionality .....	24
3.3 Non-Functional Requirements .....	27
3.3.1 Usability Requirements .....	27
3.3.3 Performance Requirements .....	29
3.3.4 Scalability Requirements .....	29
3.3.5 Portability Requirements .....	29
3.4 System Architecture .....	30
3.4.1 Logical View using Class Diagram .....	30
3.4.2 Process View .....	31
3.4.3 Development View .....	33
3.4.4 Physical View .....	34

3.4.5 Use Case View.....	35
3.5 System Evaluation Methodology .....	38
3.5.1 System Test.....	38
3.5.2 Stress and Volume Test .....	38
3.5.3 Usability Test.....	38
IMPLEMENTATION AND RESULT DISCUSSION .....	40
4.1. Overall System Implementation.....	40
4.2. Module Implementation (Interfaces).....	41
4.2.1 Model Construction .....	41
4.3. Module Implementation (Code) .....	54
4.4. System Evaluation & Result Discussion.....	54
Conclusion and future recommendations.....	56
References .....	58

## LIST OF FIGURES

Figure 1 System Overview.....	7
Figure 2 Satisfying Chips Constraint: 1 Notebook + 1 Desktop $\leq 10000$ [] .....	11
Figure 3 Continuing this process and adding in rest of the constraints. [].....	11
Figure 4 The optimal profit occurs at point X [].....	12
Figure 5 Model File built for AMPL Solver .....	13
Figure 6 Pricing Policy of the Mimosa Scheduling Software.....	18
Figure 7 Results for different solvers applied to benchmark problem instances .....	22
Figure 8 Running times of several solvers applied to benchmark test data.....	22
Figure 9 displays the number of problem instances that have been successfully solved by each solver .....	23
Figure 10 Class Object Diagram of Optimera .....	30
Figure 11 Activity Diagram of Optimera.....	31
Figure 12 Sequence Diagram of Optimera .....	32
Figure 13 Component Diagram of Optimera .....	33
Figure 14 Deployment Diagram of Optimera.....	34
Figure 15 Use Case View of Optimera .....	35
Figure 16 Use Case View Of Model Construction .....	36
Figure 17 Use Case View of Model modification .....	37
Figure 18 Menu Options to Construct Model .....	41
Figure 19 the user is prompted to name the model.....	41
Figure 20 The user adds variable by providing a variable name, type, objective coefficient, lower bound and upper bound. ....	43
Figure 21 The user adds objective function by listing down the left side linear expression consisting of variables, and sense.....	44
Figure 22 The user adds constraint by listing down the left side linear expression consisting of variables, the constraint sense and the right hand side. ....	45
Figure 23 The user can modify the variables, constraint and objective function of the model already constructed. ....	46
Figure 24 The user can save the constructed or modified model to disk.....	47
Figure 25 The user can import a model file of .lp file format .....	48
Figure 26 optimal result found.....	49
Figure 27 Infeasible model .....	50
Figure 28 Menu options for Solution Export.....	51
Figure 29 Solution exported to .XLSX .....	51
Figure 30 Solution exported to .PDF .....	52
Figure 31 Solution exported to .HTML .....	52
Figure 32 Error message displayed when user entered text into integer input field.....	54

Figure 33 Error message displayed when user tries to edit constraint when the model has no constraints.....	55
--	----



## ABSTRACT

Businesses, organizations and firms all around the world face a lot of problems that can be modelled and solved using constraint optimization [1-3] to determine the outcome of the problem [22]. Given a transformation between input and output values, described by a mathematical function  $f$ , optimization deals with generating and selecting a best solution from some set of available alternatives, by systematically choosing input values from within an allowed set, computing the output of the function, and recording the best output values found during the process. For example, the inputs can be design parameters of a motor and the output can be the power consumption. Solving these problems optimally is a hassle that requires technical knowledge of optimization techniques [40]. These techniques mathematically involve complex algorithms [44], variables and functions that are not a cup of tea for most parties trying to solve their problem.

In our Final Year Project, we are developing a .NET application, by the name of Optimera, with an intuitive user-friendly interface that provides optimal solution for all kinds of resource allocation linear programming problems that academic, government and business entities face every day. Unlike most other scheduling applications, Optimera will not solely serve as a tool to support a particular organization's scheduling and resource allocation process. Instead, it shall focus on the core challenges of resource allocation and provide a user-centric solution to those. Hence, it can be deployed in any organization for any scheduling and resource allocation from allocation of students according to an examination seating plan to allocation in elections. With Optimera you can easily import data from external systems and export your completed results back again giving you the freedom to schedule the way you want.

# **INTRODUCTION**

Optimera is a .NET solution for constraint optimization [1-3] for scheduling and resource allocation problems using linear programming solvers. It allows you to perform hands-on interaction and experimentation with optimization models. You can read models from files, perform complete or partial optimization runs on them, change parameters, and modify the models, re-optimize, and so on. You can easily import data from external systems and export your completed results back again giving you the freedom to schedule the way you want.

This section of the report contains the purpose and outline of this document, the intended audience of this document, the motivation behind the project, the objectives and scope of the proposed system and the methodology of the approach taken.

### **1.1. Document Purpose**

This report aims to provide its readers complete information regarding Optimera. All of the stages of development starting from viability and proof of concept through literature review, design, implementation, results analysis and concluding remarks have been documented in this report.

This document, therefore, ensures that both the technical and non-technical reader understands why and how services provided by the system have been implemented and under which operational constraints, the system has been designed and implemented. We have sought to provide sufficient detail to make it easy for non-technical readers to comprehend the information however technical readership will never the less find this report much easier to comprehend.

## **1.2. Intended Audience and Document Overview**

The Intended Audience of this document will be the development team, the possible client and the Supervisors Dr. M. Moazam Fraz & Dr. Mian Hamayun. Any developers who want to work further on the evolution and development of future versions of Optimera can also be benefitted.

As aforementioned, this document aims to provide its readers complete information regarding Optimera. All of the stages of development starting from viability and proof of concept through literature review, design, implementation, results analysis and concluding remarks have been documented in this report. The document is structured into five sections and two appendices. This document must be read section-wise to grasp a better understanding of the developed system. The contents of each chapter are briefly explained as follows:

### **Chapter 1: Introduction**

The introduction highlights the problem being faced in the absence of the software and underlines how this software aims to counteract the problem. The motivation of the entire project is discussed and the methodology of the approach to solving the problem.

### **Chapter 2: Literature Review**

The literature review puts light on the key concepts necessary to understand the technicalities of the project. It further points out the unmet demand of the problems currently faced which previous works in research and academia were unable to cater to and how Optimera presents a solution to these problems.

### **Chapter 3: Functionality and Design**

This section lists down the functional and non-functional requirements for Optimera gathered through scenarios, ethnography and interviewing the stakeholders of the proposed system. The section also lists down the software design specifications for Optimera gathered through modelling of the system using UML diagrams. These

software design specifications include the architectural, interface and component-level design for the system. It also specifies the application domain, system boundaries and operational constraints to be considered in implementation phase.

#### **Chapter 4: Implementation and Result Discussion**

This section explains the procedure used to implement the design and some of the challenges faced. The class diagram and code snippets are added for each module for further study of the implementation. Some problems have been modelled and solved to discuss the result of the system developed.

#### **Chapter 5: Conclusion and Future Recommendations**

This section concludes the results achieved through the development of the new system and highlights the impact on our society. It also mentions some of the possible audience of the system and how it can be deployed in the surroundings. The section ends with a brief discussion about the limitations of the system and how it can be improved in the future.

### **1.3. Problem Statement and Motivation for the New System**

Businesses, organizations and firms all around the world face a lot of problems that can be modelled and solved using constraint optimization [1-3] to determine the outcome of the problem [22]. One such example is that of university seating arrangements. The largest universities in the world may contain a hundred thousand students in a campus and arranging these students for some purposeful activity such as examination is a critical problem because the resources are limited and conditions are constrained. The seating arrangements must be conducted optimally to ensure that the examination is conducted in the best possible way. The problem outlined can be related on a small level to the seating arrangement system in our very own school SEECS where arranging around a 1000 students can become a big problem for the administration. Another example to consider is the allotment of constituencies and polling stations to more than 1 crore voters prior to the general election in Pakistan. This allotment is constrained by certain constraints and manually finding the correct allotment for all the voters is a problem faced in all the general elections.

Solving these problems optimally is a hassle that requires technical knowledge of optimization techniques [40]. These techniques mathematically involve complex algorithms [44], variables and functions that are not a cup of tea for most parties trying to solve their problem.

In section 2.1.2 “Modelling using Linear Programming [42]”, we have explained the graphical method of solving a problem bounded by constraints [45] and highlighted the level of difficulty with increasing problem size. It can be clearly seen that manually generating the optimized solution for all such problems is time-consuming and labor-intensive when the problem size increases. The second approach of solving these problems is using the solver libraries available in the market that unfortunately, an individual without ample programming skills fails to take advantage of. The last approach is using commercial software products but these are very expensive to buy and deploy. Thus, it can be clearly seen that there exists a gap for

softwares that can model and solve these constraint optimization problems using linear programming given the resources in hand and constraints that are to be considered.

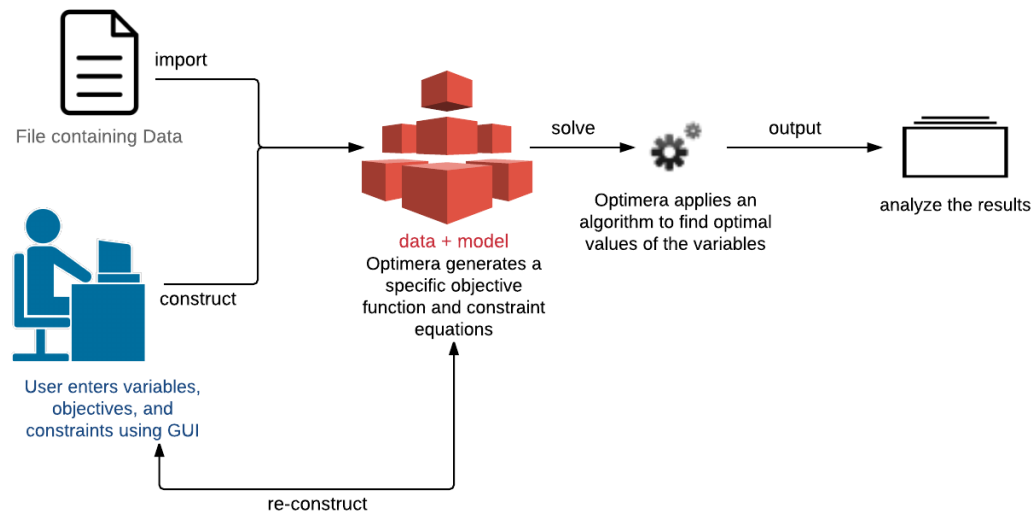
The system developed as our final year project is a software solution that provides optimal solution for all kinds of resource allocation linear programming problems of a large scale i.e. 64,000 entities. Our software also provides a user-friendly interface that simply takes constraints and resources as input values and output the values that result in the best available result. This would ensure that the end-user be free of all the unnecessary mathematical complexities.

#### **1.4. Product Overview**

Our project Optimera consists of developing a Constraint Optimization Software that maps real life problems of resource allocation and scheduling into mathematical functions and generates the optimal solution from some set of available alternatives, by systematically choosing input values from within an allowed set, computing the output of the function, and recording the best output values found during the process.

It allows you to perform hands-on interaction and experimentation with optimization models. You can formulate a model - the abstract system of variables, objectives, and constraints that represent the general form of the problem to be solved; collect data that define a specific problem instance; and generate a specific objective function and constraint equations from the model and data. You can then solve the problem instance by running a solver to apply an algorithm that finds optimal values of the variables and analyze the results. These solver libraries deployed to cater to all kind of linear programming problems, use various algorithms such as Simplex Algorithm [25], parallel barriers [8] etc., allowing users to state their toughest problems as mathematical models, and then automatically considering billions or even trillions of possible solutions to find the best one.

You can also modify the model by changing constraints and variables and re-optimize the model. The following diagram illustrates show the system interacts with its external environment and the way the system is put to use:



**Figure 1 System Overview**

With Optimera you can easily import data from external systems and export your completed results back again giving you the freedom to schedule the way you want. Unlike most other scheduling applications, Optimera will not solely serve as a tool to support a particular organization's scheduling and resource allocation process. Instead, it shall focus on the core challenges of resource allocation and provide a user-centric solution to those. Hence, it can be deployed in any organization for any scheduling and resource allocation from allocation of students according to an examination seating plan to allocation in elections.

# LITERATURE REVIEW

This section puts light on the key concepts necessary to understand the technicalities of the project. Next, it enumerates the solutions that are provided already in the market to solve problems in the domain of scheduling and resource allocation and discuss what their shortcomings are. We point out the unmet demand of the problems currently faced which previous works in research and academia were unable to cater to and how Optimera presents a solution to these problems. We shall also do a technology review of the solver libraries available that were available for our solution.

## 2.1. Key Concepts

### 2.1.1. Constraint Optimization

Say a function  $f(x_1, x_2 \dots x_n)$  needs to be maximized or minimized with respect to the variables  $x_1, x_2 \dots x_n$  in the presence of constraints on those variables. This function can be a cost function or energy function or utility function.

**MAXIMIZE**       $f(x_1, x_2 \dots x_n)$  <- Objective function

**SUBJECT TO**

Inequality constraints ->

$$g_1(x_1, x_2 \dots x_n) \leq b_1 \dots g_k(x_1, x_2 \dots x_n) \leq b_k$$

Equality constraints ->

$$h_1(x_1, x_2 \dots x_n) = c_1 \dots h_m(x_1, x_2 \dots x_n) = c_m$$

This process of optimizing an objective function with respect to some variables in the presence of constraints on those variables is known as constraint optimization. “In mathematics, a constraint is a condition of an optimization problem that the solution must satisfy. There are several types of constraints—primarily



equality constraints, inequality constraints, and integer constraints. The set of candidate solutions that satisfy all constraints is called the feasible set.” [47]

“In constrained optimization, Constraints can be either hard constraints which set conditions for the variables that are required to be satisfied, or soft constraints which have some variable values that are penalized in the objective function if, and based on the extent that, the conditions on the variables are not satisfied.” [47]

### **2.1.2. Modelling with Linear Programming**

Linear programming or linear optimization is an approach to find the best outcome (such as maximum profit or lowest cost) in a mathematical model of a linear objective function, subject to linear equality and linear inequality constraints. Let’s consider an example from CMU Course [34] to understand how linear programming is applied to solve a model.

A company makes decisions about their products for a quarterly year basis. Their two products are notebooks and desktops. The company would like to know the production units of each product for the respective quarterly period in order to maximize profit for the quarter. There is a limitation on the rate of production:

1. The company is supplied with only a maximum of 10,000 chips. Each computer requires a chip so the value of 10,000 acts as a constraint.
2. Each computer has a requirement for memory which comes in the form of 16Mb chip sets. A desktop requires 2 chip sets (32 Mb) whereas a notebook requires 1 chip set (16 Mb). The company already has a stock of 15,000 chip sets which it can use for the coming months.
3. A notebook takes 4 minutes to assemble whereas a desktop requires 3 minutes to assemble. This gives us the assembly time constraints. Each quarter has 25,000 minutes of assembly time.

The profits generated by the two computers differ. The notebook produces a profit of 750\$ whereas a desktop generates a profit of 1,000\$.

To solve this problem using linear programming, we shall form a model consisting of decision variables, objective, and constraints. The decision variables represent unknown decisions to be made, unlike the problem data that is provided. For this problem, we shall represent the decision variables as the number of notebooks to produce and the number of desktops to produce, i.e. *Notebook and Desktop*. Our objective is to *Maximize* the profit function  $750 \text{ Notebook} + 1000 \text{ Desktop}$ . The constraints limiting the feasibility region for this function shall be of four types: *Processing chips, assembly, memory sets, and non-negativity*.

Decision Variables: Notebook, Desktop

Objective Function: Maximize  $750 \text{ Notebook} + 1000 \text{ Desktop}$

Constraints:

Chips:  $1 \text{ Notebook} + 1 \text{ Desktop} \leq 10000$

Memory sets:  $1 \text{ Notebook} + 2 \text{ Desktop} \leq 15000$

Assembly time:  $4 \text{ Notebook} + 3 \text{ Desktop} \leq 25000$

Notebook  $\geq 0$

Desktop  $\geq 0$

This gives us the complete model of this problem. Now, let's find the solution of the problem defined by the feasible region of intersection of finitely many half spaces, each of which is defined by a linear inequality. The model can be represented with the two variables being on the x and y axis. The possibilities are represented by the entire graph. The lines represent the constraints while the feasibility is represented by the area bounded by the lines. Figure 2 illustrates this with the processing chips constraint.

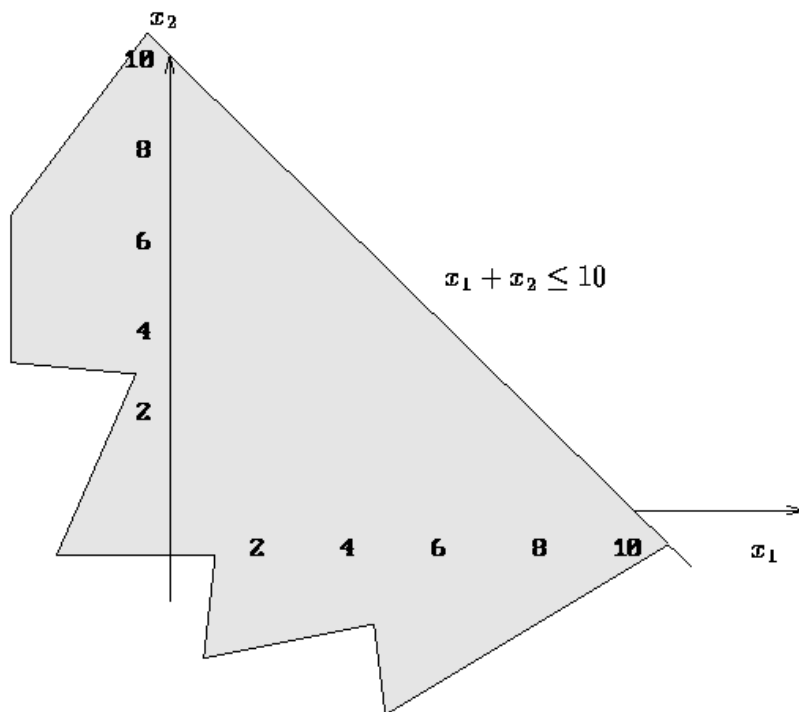


Figure 2 Satisfying Chips Constraint: 1 Notebook + 1 Desktop  $\leq 10000$  [34]

The resulting feasible region is the intersection of the feasible region for each constraint as each constraint must be satisfied. This is depicted in figure 3.

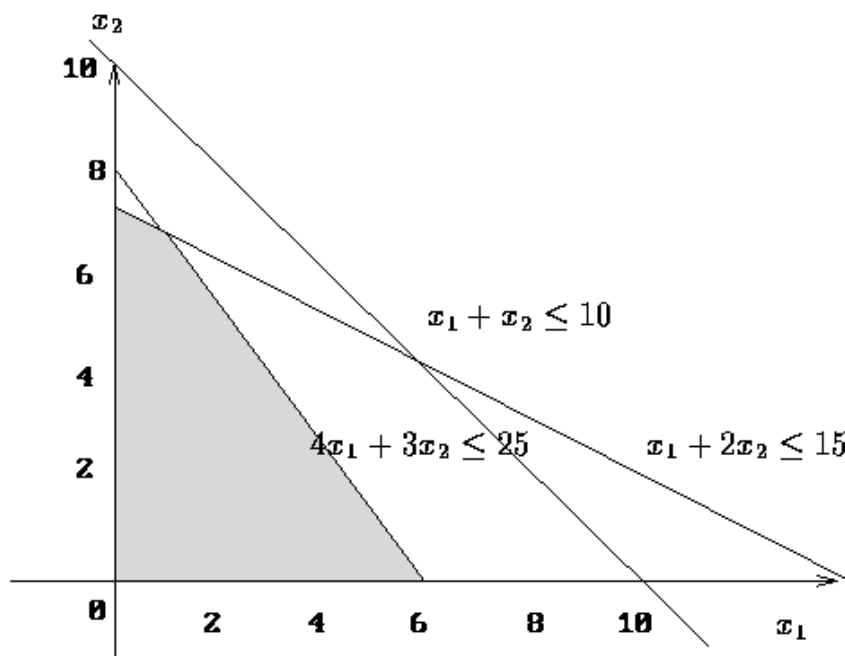


Figure 3 continuing this process and adding in rest of the constraints. [34]

The objective function is graphed by drawing iso-profit lines i.e. lines where the profit is the same. Since we have to maximize the profit, we push the iso-profit lines outwards to the point where any more increase will result in infeasibility. Clearly, the optimal profit occurs at point X where  $x_1 = 1$  and  $x_2 = 7$ , reflecting that the optimal decision is to produce 1,000 notebooks and 7,000 desktops, for a profit of \$7,750,000.

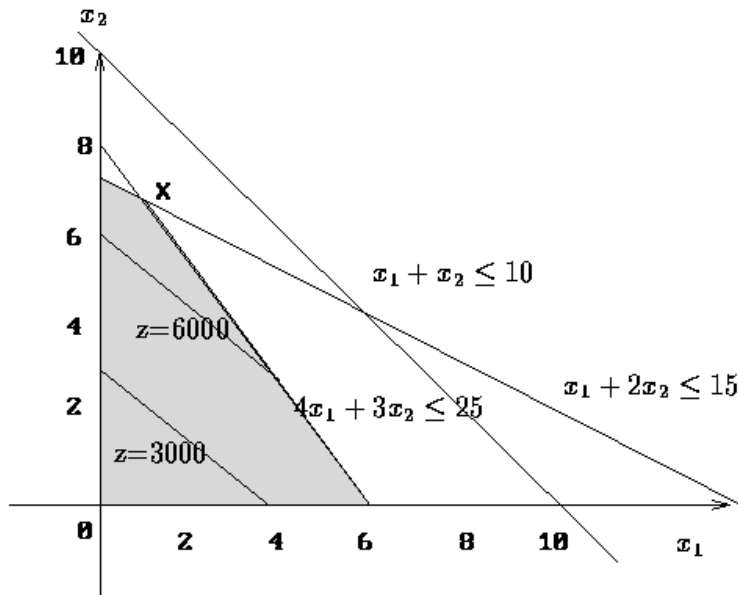


Figure 4 the optimal profit occurs at point X [34]

This graphical method is limited to linear programs with 2 variables, and gets tedious and error-prone if we try to increase the number of variables and constraints in the problem. A second approach to modelling and solving using linear programming is using AMPL-Based [37] or other linear programming language based solver libraries. Unfortunately, with these solvers available in the market, you need to have knowledge about the programming language syntax. Take AMPL Input for example, you need to build model file as shown below.

```

set batch;
set batch_subject{batch} within subject;

set day_time within{day, time};
set course_conflict{subject} within subject; # with respect to student
param course_strength{subject};

var X{subject, day_time} binary;

minimize subject_per_day: sum{s in subject, (d,t) in day_time} X[s,d,t];

#All subjects are scheduled
subject to schedule{s in subject}:
    sum{(d,t) in day_time} X[s,d,t]=1;

#Subject and its conflicting subjects cannot be scheduled on same time.
subject to avoid_subject_conflict{(d,t) in day_time, s in subject}:
    X[s,d,t] + (sum{c in course_conflict[s]} X[c,d,t])/(sum{c in course_conflict[s]}1)<=1;

#Subject to Batch conflict
subject to batch_conflict{b in batch,d in day}:
    sum{s in batch_subject[b], t in time}
        X[s,d,t]<=2;

#Students scheduled at a given time are less than seating capacity
subject to seating_capacity{(d,t) in day_time}:
    sum{s in subject} X[s,d,t]*course_strength[s]<=680;

```

Figure 5 Model File built for AMPL Solver

### 2.1.3. How do solvers work?

Solvers are mathematical softwares that solve mathematical problems automatically once inputs are provided. Solvers may be available in the form of stand-alone programs or software libraries that can be used by external programs. The problems presented may be linear problems, mixed integer problems [28], quadratic problems [46], quadratic mixed integer problems [43] etc. Different solvers may have the ability of solving types of problems. A variety of different solvers are discussed in Section 2.3.

The inputs provided are generally in the form of variables that are constrained by a number of constraints applied. The objective is to maximize or minimize an equation of these variables. Let's consider a simple example:

Variable x, y and z are inputted in the solver where the objective function is to maximize the value of  $x+2y-z$  given the constraints that  $5>x$ ,  $y<5$  and  $z>3$ . After applying initial heuristic values to x, y and z the solver iterates the different

possibilities of values for the variables while slowly converging to the correct solution. The % of uncertainty of optimal solution decreases at each iteration to finally give the optimum solution. If extremely complex equations are provided or if the equation is contradictory then then the solver may yield an infeasible solution. Detailed workings of the solvers are beyond the scope of this project as the solvers are merely the tools used in an external application to help achieve the objective of this project.

## **2.2. Available Market Solutions**

### **2.2.1 CourseLeaf Section Scheduler (CLSS):**

CLSS [6] is a web based application with visualizing tools for schedule distribution, course handling, information system, room scheduling and format printing. With CLSS, faculty and staff can coordinate across multiple departments and offerings to build a schedule of classes that is consistent across campus. CLSS saves resources and enhances efficiency as according to a report by a leading company, CLSS decreased data entry by their registrar's office by a value of up to 90%, claiming that the number of changes required by the staff had decreased by a value of 180,000.

### **2.2.2 Event Management Systems (EMS):**

This is the mission statement of Event Management Systems [9]: “For over 25 years, our mission has been to provide facilities of all types with comprehensive, best-of-breed software solutions for room scheduling, meeting services management, web calendaring and online event registration/surveys. In doing so, we strive to deliver insightful professional services, powerful product integrations and award-winning technical support so that the end result is organizations that work more efficiently, communicate more effectively and maximize the value of their resources.”

EMS handles hundreds of thousands of meeting and events annually across more than 2.5 million people and 75 countries. EMS provides event, classroom and exam scheduling as well as an automatic procedure for room assignment. EMS campus uses an algorithm that determines the most effective course schedules

according to the students and instructors data. The different functionalities stated above are provided by different softwares and must not be mistaken for single software.

### **2.2.3 Time Tabler:**

Time Tabler [43] is a teacher-scheduling software that can also be used to schedule staggered breaks, manage school events and set consortium dates. The self-checking capability allows for avoiding conflicting schedules such as allocation of the same room to two teachers of different subjects at the same time. The scheduling process is carried out automatically or manually. Constraints may be relaxed to provide a feasible solution if the software states that the solution is infeasible.

### **2.2.4 EZnet Scheduler:**

EZnet Scheduler [10] is an online appointment scheduling program. EZnet scheduler uses voice reminders, custom forms and scheduling based on three levels to enhance efficiency in scheduling routines.

### **2.2.5 Mimosa Scheduling Software:**

Mimosa [36] claims to be the world's most versatile scheduling software. Mimosa aims to provide generic solutions instead of providing solutions for a particular organization. Hence Mimosa is used by a wide variety of organizations including schools and businesses.

Mimosa makes maintaining timetables easier. Using a collection of optimization algorithms, Mimosa schedules efficiently and effectively.

### **2.2.6 Standard Solver in MS Excel:**

The Excel Standard Solver is part of a suite of commands sometimes referred to as what-if analysis tools. It allows finding an optimal (maximum or minimum) value for a formula in one cell called the objective cell subject to constraints on the values of other formula cells on a worksheet. Solver in Excel has a limit of 200 decision variables and 100 constraints. You can upgrade to the premium version for a monthly fees to get a limit of 2,000 decision variables for linear problems, and 500 variables for nonlinear problems, and a limit of 1,000 constraints for linear problems, and 250 constraints for nonlinear problems. This still makes Excel Standard Solver less powerful than other solvers and the Excel users also require technical knowledge of MS Excel and Spreadsheets.



### 2.2.7 Concluding Remarks:

<b>Problems:</b> ✓ MONEY ✓ SPECIALIZED	<b>Specialized Commercial Software for scheduling:</b> CourseLeaf Section Scheduler(CLSS) Event Management Systems (EMS) Time Tabler EZnet scheduler Mimosa Scheduling Software
<b>Problems:</b> ✓ PROGRAMMING CHALLENGE	<b>AMPL-Input based Solvers:</b> BLMVM, BiqMac, Knitro, PATHPGAPack, Pswarm, scip, CPLEX, MINTO, MOSEK, proxy, qsopt_ex, LANCELOT, LOQO, LRAMBO
<b>Problems:</b> ✓ LESS POWERFULL THAN SOLVERS ✓ STILL REQUIRES TECHNICAL EXPERTISE	<b>Excel Solver</b>

The software listed above and even those off the list are mostly scheduling software which is not exactly the same as constraint optimization software. The only purpose of scheduling software is to provide the best available schedule whereas the goal of a constrained optimizing software is to find the optimal solution to a constrained problem be it a scheduling problem (timetable scheduling in university) or not (constituency allotment to voters). Furthermore the solutions provided by the software in the market are not to resolve multiple types of generic optimizing problems; rather they aim to solve specified problems such as appointment scheduling, final exam scheduling, seating arrangement etc. The only software found to be solving generic problems is the mimosa scheduling software but then again it only solves problems that are time oriented and related to scheduling only. Furthermore the prices of these software's are quite high and not easily affordable by institutions. The table provided below shows the expensive pricing policy of the Mimosa Scheduling Software [36].

### Mimosa Site Licence Prices (€)

Students	Price (€)	Students	Price (€)
< 800	€ 800	4,500	€ 4,500
800	€ 800	5,000	€ 5,000
1,000	€ 1,000	5,500	€ 5,500
1,500	€ 1,500	6,000	€ 6,000
2,000	€ 2,000	6,500	€ 6,500
2,500	€ 2,500	7,000	€ 7,000
3,000	€ 3,000	7,500	€ 7,500
3,500	€ 3,500	8,000	€ 8,000
4,000	€ 4,000	> 8,000	€ 8,000

Figure 6 Pricing Policy of the Mimosa Scheduling Software [36]

With excel solver being the most suitable solution but one which has a limited number of variable and constraints size, we are left with a gap in the arena user-friendly optimization software. This gap needs to be filled by a software solution that can model and solve mathematical models of a larger problem size and that can provide a user intuitive graphical interface.

## **2.3. Technology Review**

In this section, we shall review some of the mathematical optimization solvers available in the market and evaluate their feasibility of use in our project by studying the performance comparison of different (mixed integer) linear solvers by Hans Mittelmann [19]. Various instances of linear problems using serial and parallel solvers are benchmarked using real world test cases from the Netlib repository [19].

### **2.3.1. Open Source Solvers**

#### **2.3.1.1 GLPK (GNU linear programming kit)**

GLPK [18] is an open source software written in ANSI C that's intends to solve problems of the Linear Programming and Mixed Integer Programming domains as well as other related domains. GLPK is organized in the form of a callable library. It supports a subset of the AMPL language known as the GNU MathProg modeling language. Some of the algorithms used include primal and dual simplex methods [31], branch-and-cut method and primal-dual interior-point method [25].

#### **2.3.1.2 LP solve**

LP solve [32] is another example of an open source solver that can be used to solve MIP and LP problems. Along with that the LP solve also solves semi-continuous and special ordered sets (SOS) models. LP solve is written in ANSI C and can be compiled on multiple operating systems such as WINDOWS and Linux.

#### **2.3.1.3 CLP (Coin-or lp)**

CLP [5] is a solver written in C++ to tackle linear optimization problems and is used in many other projects within Coin-OR such as SYMPHONY which is used for solving MIP problems, CBC for solving linear programming problems based on branch and cut algorithm or BCP which is a framework for constructing parallel branch-cut-price algorithms for MIP programs. CLP is used as a callable library although it is possible to build a stand-alone execution version.

#### **2.3.1.4 SCIP**

SCIP [39] is currently one of the fastest non-commercial solvers for mixed integer programming and mixed integer nonlinear programming. SCIP can be operated as a callable library in C or a standalone solver which is also a framework for constraint integer programming employing the branch-cut-and-price method [38].

#### **2.3.1.5 SoPlex**

SoPlex [41] is a Linear Programming model solver based on the revised simplex algorithm which is implemented in C++. SoPlex is open source can be used as a standalone solver or as a callable C++ library.

### **2.3.2. Commercial LP Solvers**

#### **2.3.2.1 Gurobi**

Gurobi Optimizer [16] is a modern solver for LP, QP, QCP, and MIP (MILP, MIQP, and MIQCP) problems. The solver has been written in C and can be deployed on WINDOWS and Linux. Gurobi fully exploits parallelism and uses advanced MIP cutting plane routines, MIP heuristics and makes use of simplex and barrier algorithms. Gurobi has support from Object-oriented interfaces for C++, Java, .NET, and Python, Matrix-oriented interfaces for C, MATLAB, and R and links to standard modeling languages such as AMPL, GAMS, AIMMS, and MPL.

#### **2.3.2.2 Ibm Ilog Cplex**

Cplex [24] is a commercial solver designed to solve large MILP problems. CPLEX exploited the simplex method as written in C. Cplex is not restricted to simplex only and exploits many other optimization techniques along with providing interfaces to languages other than C. IBM has actively played a part in the development of Cplex.

#### **2.3.2.3 Fico Xpress**

Fico Xpress is designed to solve MILP problems. Along with providing a command line interface Xpress also provides several callable library API's for several programming languages and can be deployed on commonly used computer platforms.

### **2.3.3 Performance Comparison**

We shall compare the performance of the mentioned solvers by studying the performance comparison of different (mixed integer) linear solvers by Hans Mittelmann [19]. Various instances of linear problems using serial and parallel solvers are benchmarked using real world test cases from the Netlib repository on this webpage.

	running time	instances solved	solved (%)
CBC	10.20	41	47.13
CPLEX	1.45	73	83.91
GLPK	22.11	3	3.45
GUROBI	1.00	77	88.51
LP_SOLVE	19.40	5	5.75
SCIP-C	3.76	63	72.41
SCIP-L	6.40	52	59.77
SCIP-S	5.33	57	65.52
XPRESS	1.29	74	85.06

Figure 7 Results for different solvers applied to benchmark problem instances [19]

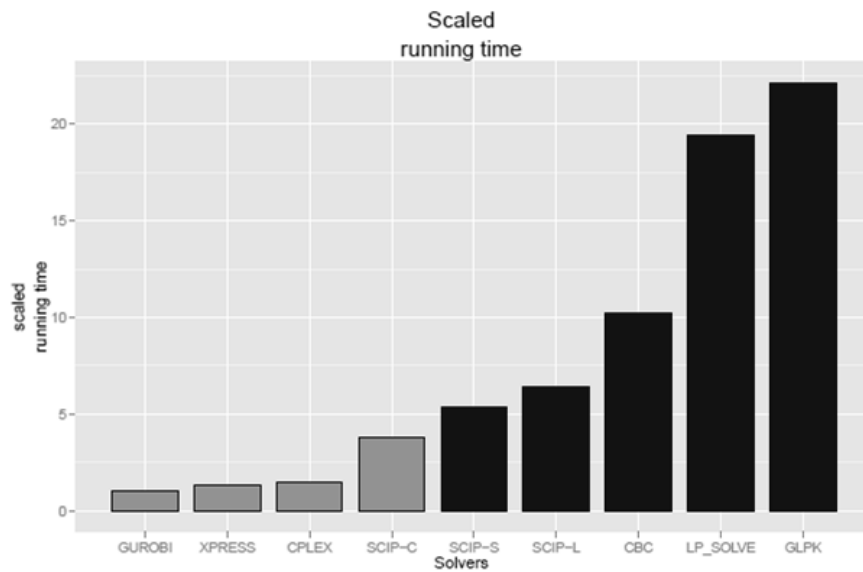


Figure 8 Running times of several solvers applied to benchmark test data [19]

Figure 7 shows the main results. In this table the first column shows the solver used. In the second column the running times are shown for each solver. The maximal running time was limited to 60 minutes. For cases in which a solver was not able to find an optimal solution within one hour, the time limit was used for further calculations. Finally, the times were modified in a way that the fastest solver

(GUROBI in this example) was scaled to 1. Figure 8 displays the performance with respect to running times. In this case smaller values are of course better.

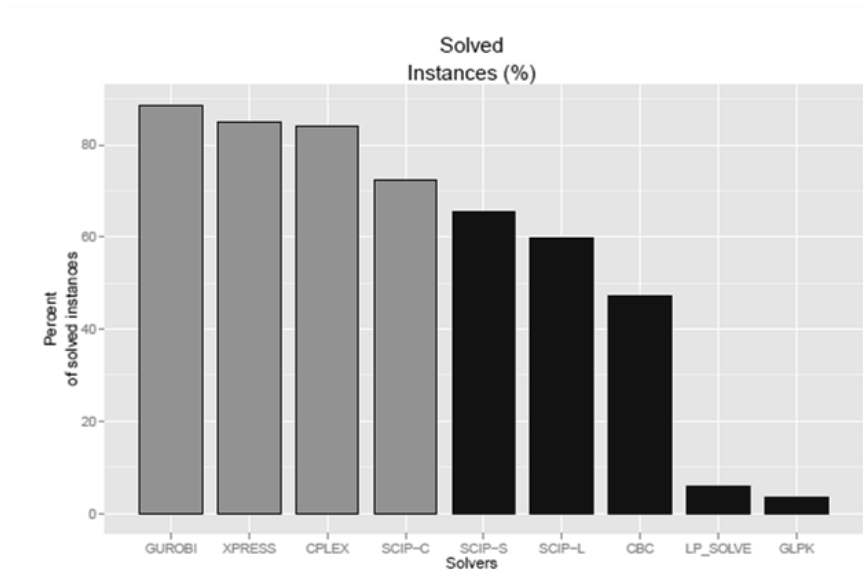


Figure 9 displays the number of problem instances that have been successfully solved by each solver [19]

The third column of figure 7 displays the number of problem instances that have been successfully solved by each solver and in the last column the corresponding percentage is listed. This information is shown graphically in Figure 9 in which higher percentages equal better performance. Figures 8 and 9 clearly show that commercial solvers (displayed in green) outperform open source solvers not only in terms of running time but also with respect to the percentage of solved problem instances when the results of all 87 problem instances are used.

# **FUNCTIONALITY AND DESIGN**

This section lists down the functional and non-functional requirements for Optimera gathered through scenarios, ethnography and interviewing the stakeholders of the proposed system. The section also lists down the software design specifications for Optimera gathered through modelling of the system using UML diagrams. These software design specifications include the architectural, interface and component-level design for the system. It also specifies the application domain, system boundaries and operational constraints to be considered in implementation phase.

## **3.1 Product Functionality**

Optimera is a Desktop application that provides optimal solution using mathematical linear optimization solvers for all kinds of resource allocation problems that academic, government and business entities face every day. It has the following features:

### **3.1.1 User Centric Solution**

It provides a user-centric solution for any scheduling and resource allocation problem of a large scale i.e. 64,000 entities. Unlike most other scheduling applications, Optimera shall not solely serve as a tool to support a particular organization's scheduling and resource allocation process. Instead, it shall focus on the core challenges of resource allocation and provide a user-centric solution to those. As a result it can be used in any organization for any scheduling and resource allocation problem ranging from the allocation of students according to an examination seating plan to the allocation of dietary resources given a minimum budget.



### **3.1.2 Intuitive User Friendly Interface**

The system provides an intuitive user-friendly interface that simply takes constraints and resources as input values and output the values that result in the best available result. This would ensure that the end-user be free of all the unnecessary mathematical complexities. It shall also provide intuitive tools that allow all individuals from different organizational backgrounds to easily understand and learn the system working.

### **3.1.3 Model Construction**

Optimera provides translation of the real life problem into mathematical model comprising of a specific objective function, variables and soft and hard constraints. The interface prompts the user to construct a model by entering variables, objectives, and constraints that represent the general form of the problem to be solved. The system collects data that define the specific problem instance, and generates a specific objective function and constraint equations from the model and data.

### **3.1.4 Model Modification**

Optimera provides allows the user to modify his/her model variables, constraints and objective function to satisfy changing needs. This enables the user to reconstruct a model after analyzing the optimization of a previous model.

### **3.1.5 Saving Model to Disk**

User shall save the constructed model in an .lp model file format for later retrieval.

### **3.1.6 Importing Model from Desk**

User shall load a model by importing an .lp model file.

### **3.1.7 Optimization of Model Using Solver Libraries**

Optimera shall perform optimization using solver libraries and algorithms. The problem instance is solved by running a solver to apply an algorithm that finds optimal values of the variables.

### **3.1.8 Export of Optimization Solution**

User shall export the optimization results to external disk in various file formats such .sol, XLSX, PDF and HTML.

### **3.1.9 Visualization of Model and Optimization Status**

Optimera provides visualization of the output using filters and customized reporting tools.

## **3.2 Design Considerations**

The system is a Windows Desktop Application. It shall be built in .NET Framework using Gurobi Library [17] for optimization algorithms. Each model shall be represented as an object of class GRBModel, consisting of a set of decision variables (objects of class GRBVar), a linear objective function on those variables (specified using GRBModel.SetObjective), and a set of constraints on these variables (objects of class GRBConstr, GRBQConstr, or GRBSOS). Each variable shall have an associated lower bound, upper bound, and type (continuous, binary, etc.). Each linear constraint shall have an associated sense (less-than-or-equal, greater-than-or-equal, or equal), and right-hand side value.

Linear constraints shall be specified by building linear expressions (objects of class GRBLinExpr), and then specifying relationships between these expressions (for example, requiring that one expression be equal to another).

The optimization model could be specified all at once, by loading the model from a file (using the appropriate GRBModel constructor), or built incrementally, by first constructing an empty object of class GRBModel and then subsequently calling GRBModel.AddVar or GRBModel.AddVars to add additional variables, and GRBModel.AddConstr or GRBModel.AddQConstr to add additional constraints. Models shall be dynamic; that is the user could always add or remove variables or constraints and re-optimize the model.

## **3.3 Non-Functional Requirements**

### **3.3.1 Usability Requirements**

Optimera is intended to be used by non-technical personnel with little or no experience with the functioning of solvers. Thus Optimera uses windows forms in C# along with advanced graphics libraries to provide easy, aesthetically pleasing, user friendly interfaces that support data input and well defined reporting. The usability

will be tested by performing end-user testing. Random people will be asked to use the software and read/build a model. The following criteria must be fulfilled:

- a. The average number of error per minute (1st use) shall be  $\leq 1$
- b. The average number of error per minute (2nd use) shall be  $\leq 0.1$
- c. The average number of errors in user's explanation of report shall be  $< 1$

### **3.3.2 Efficiency Requirements**

Our motto is to enhance efficiency by reducing complexity, cost and administrative effort by increasing productivity. Therefore it is vital that the system we are developing be efficient. The models we choose to create and solve are automatically fine tuned to the settings that shall solve the model in the most efficient manner. Linear programming problems are automatically detected and solved.

- a. Users need not worry about memory management as it is handled by garbage collector. No part of memory will be leaked during or after use of this program as the entire objects initiated are swept away by the automatic garbage collector.
- b. The solver shall terminate optimally when the relative gap between the lower and upper objective bound is less than  $1e-4$  times the upper bound.
- c. The solver shall terminate optimally when the absolute gap between the lower and upper objective bound is less than  $1e-10$  times the upper bound.
- d. Reduced costs shall be smaller than  $1e-6$  in the improving direction in order for a model to be declared optimal.

### **3.3.3 Performance Requirements**

It is important that Optimera performs up to the indicated level. The solutions provided by Optimera may affect the use of resources on a massive scale that affects the lives of a thousand people. Hence it is necessary that Optimera's optimal solution to the problem is also the universally optimal solution.

- a. All constraints must be satisfied to a tolerance of  $1e-6$ .
- b. The average error gap in the final objective solution after the iterations have run completely without interruption must be  $< 0.5\%$ .

### **3.3.4 Scalability Requirements**

The range of size of problems that Optimera shall solve is huge.

- a. The maximum number of variables that can be handled must be at least 64,000.
- b. The maximum number of constraints that can be handled must be at least 60,000.
- c. The transition between small problems and big problems need to be seamless with the only difference being the number of iterations that can rise up to a value of more than 1 billion or even 1 trillion.

### **3.3.5 Portability Requirements**

Optimera must be portable across all windows stand-alone computers that support the .NET version 4.5.

### 3.4 System Architecture

In the following pages, the 4+1 view Architecture model is used to give an understanding of how and why the system is to be decomposed, and how the individual parts work together to provide the desired functionality. It separates the different aspects of a software system into different views of the system as follows:

#### 3.4.1 Logical View using Class Diagram

The logical view of the Optimera is modelled using the following class diagram to show the key abstractions in the system.

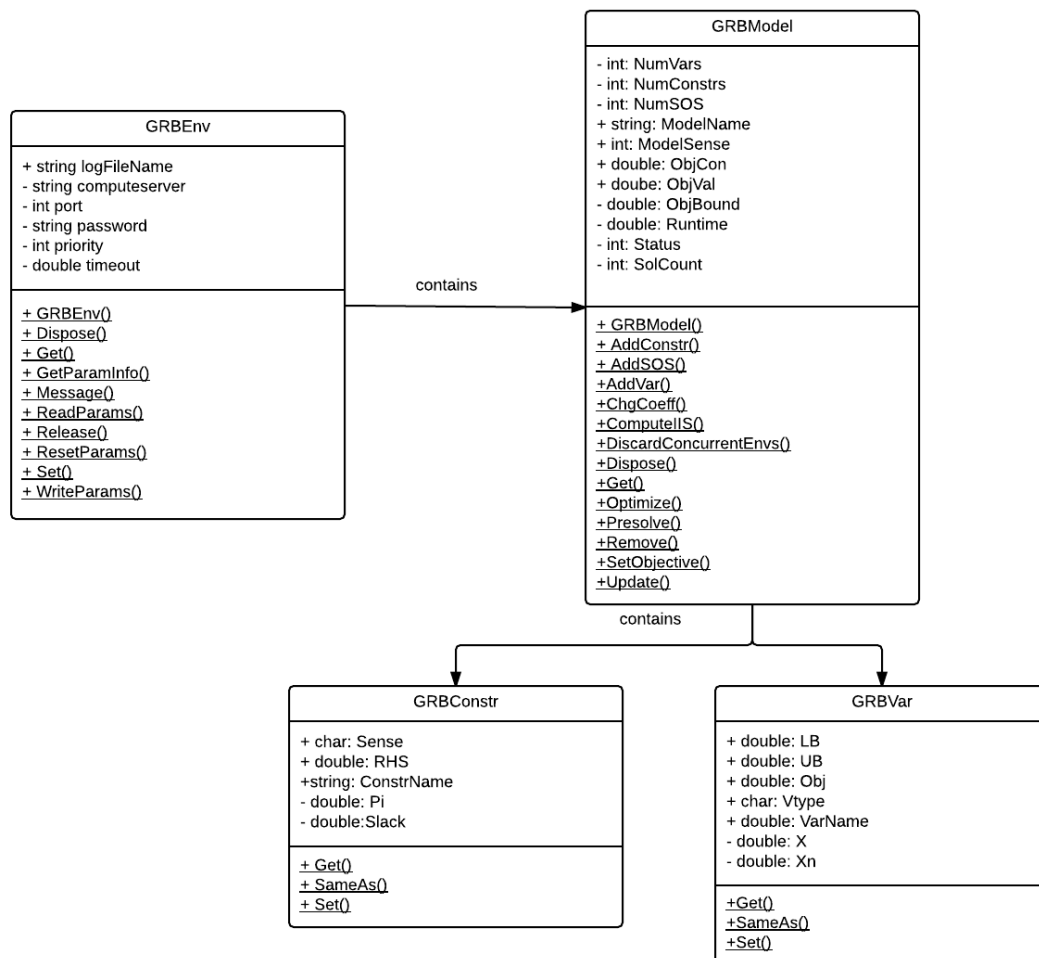


Figure 10 Class Object Diagram of Optimera

### 3.4.2 Process View

The process view of the Optimera is modelled using activity and sequence diagrams to show how, at run-time, the system is composed of interacting processes.

#### Activity diagram

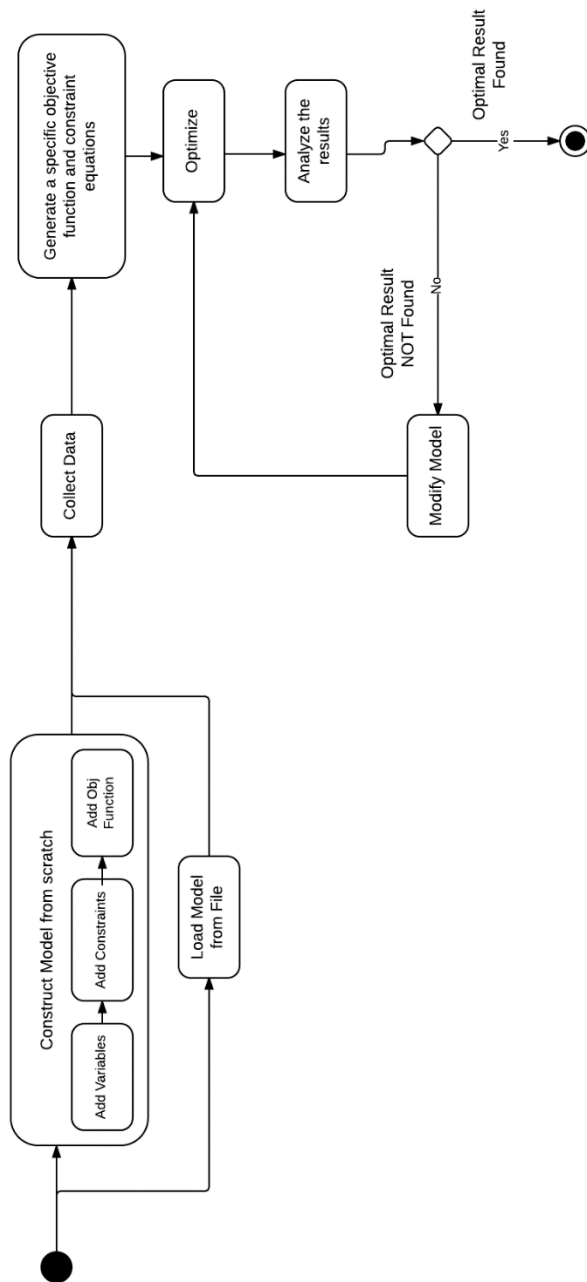


Figure 11 Activity Diagram of Optimera

## Sequence diagram

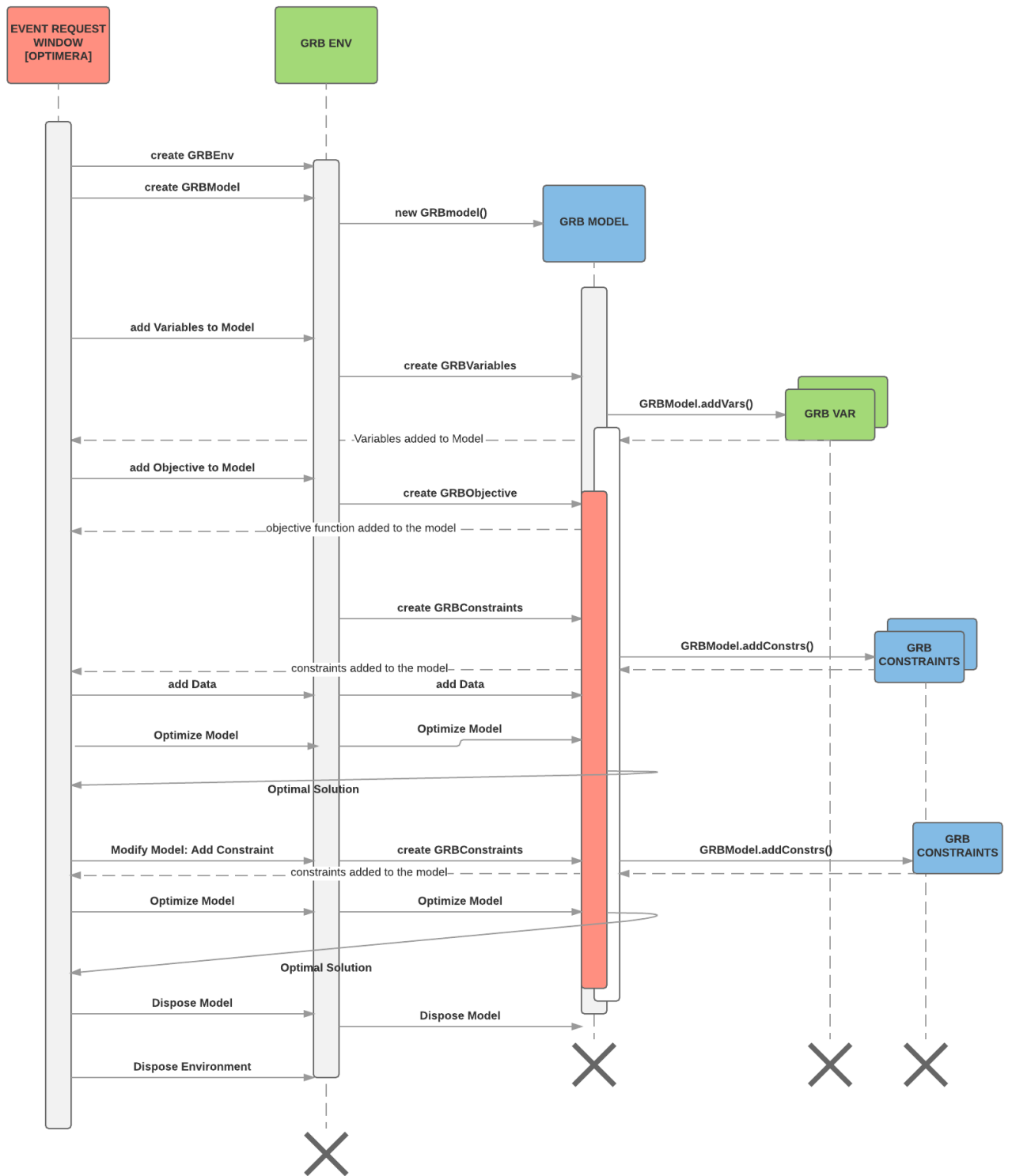


Figure 12 Sequence Diagram of Optimera



### 3.4.3 Development View

The development view of the Optimera is modelled using component diagrams to show how the software is decomposed for development.

#### Component diagram

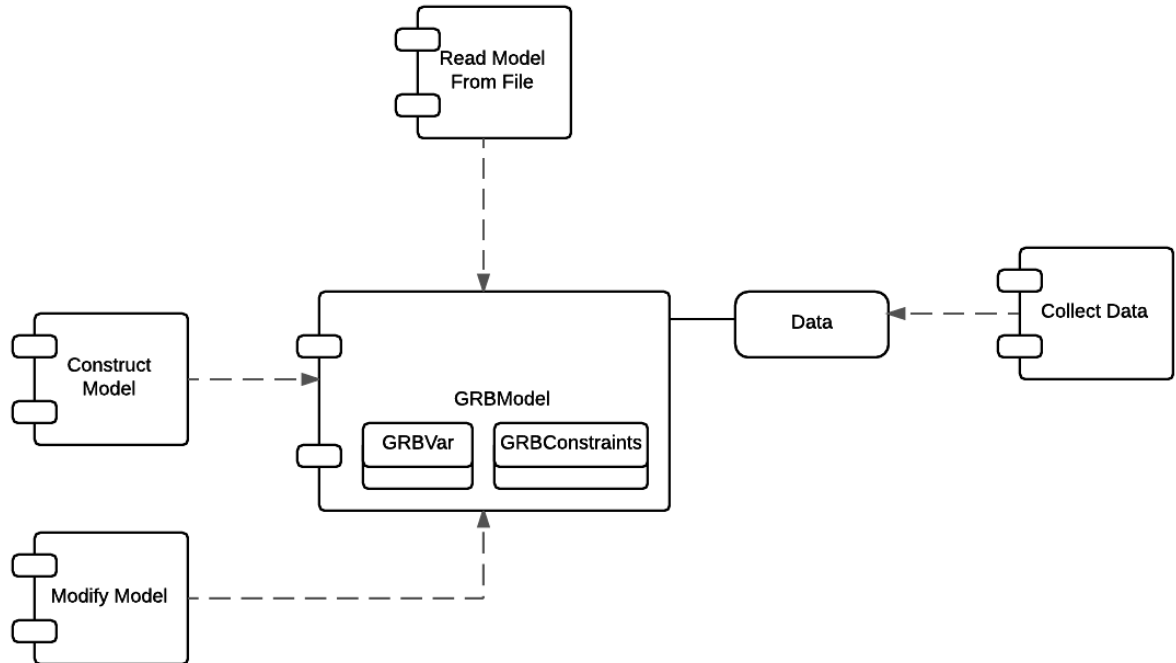


Figure 13 Component Diagram of Optimera

### 3.4.4 Physical View

The physical view of the Optimera is modelled using a deployment diagram to show the system hardware and how software components are distributed across the processors in the system.

#### Deployment diagram

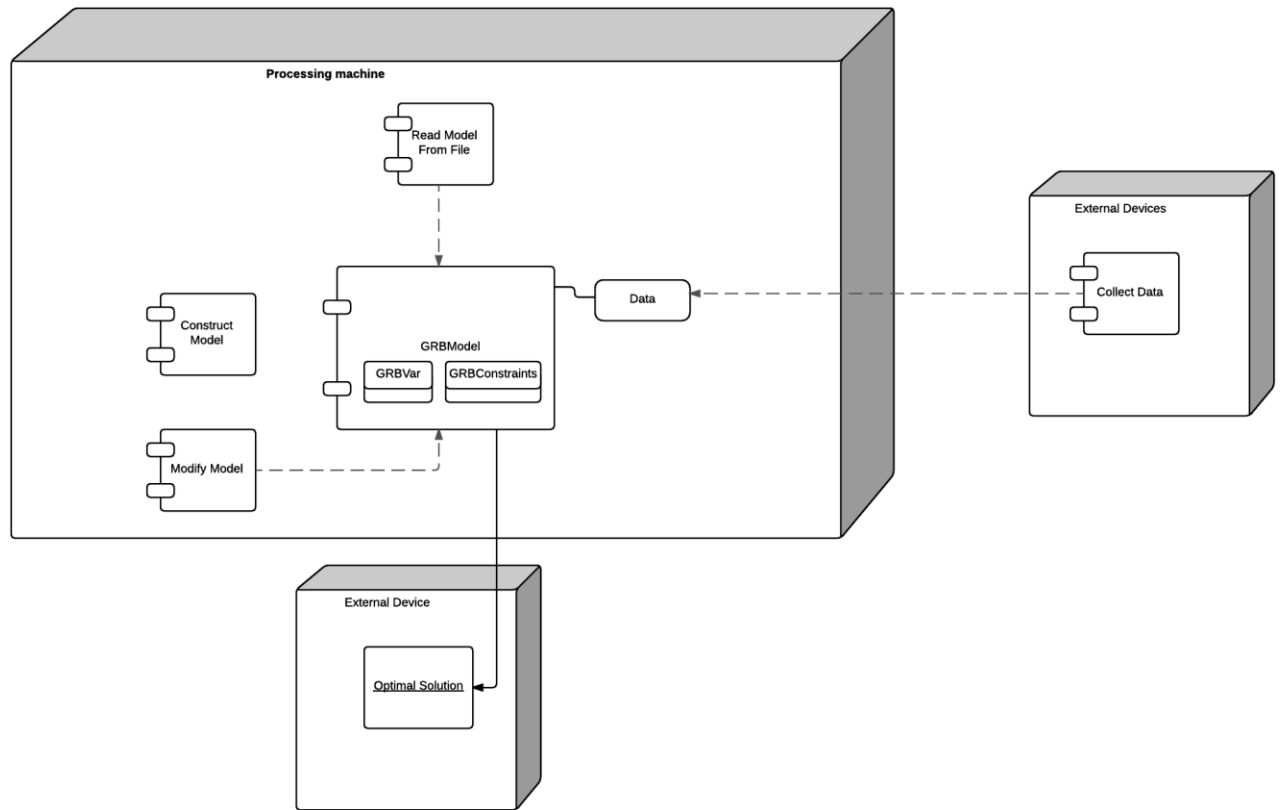


Figure 14 Deployment Diagram of Optimera

### 3.4.5 Use Case View

This view describes the functionality of the system from the perspective from outside world. It contains diagrams describing what the system is supposed to do from a black box perspective. The following use case diagram encapsulates the entire system and all possible actors. Each of these use cases have been defined further below in the SRS document.

#### Use Case View of Optimera

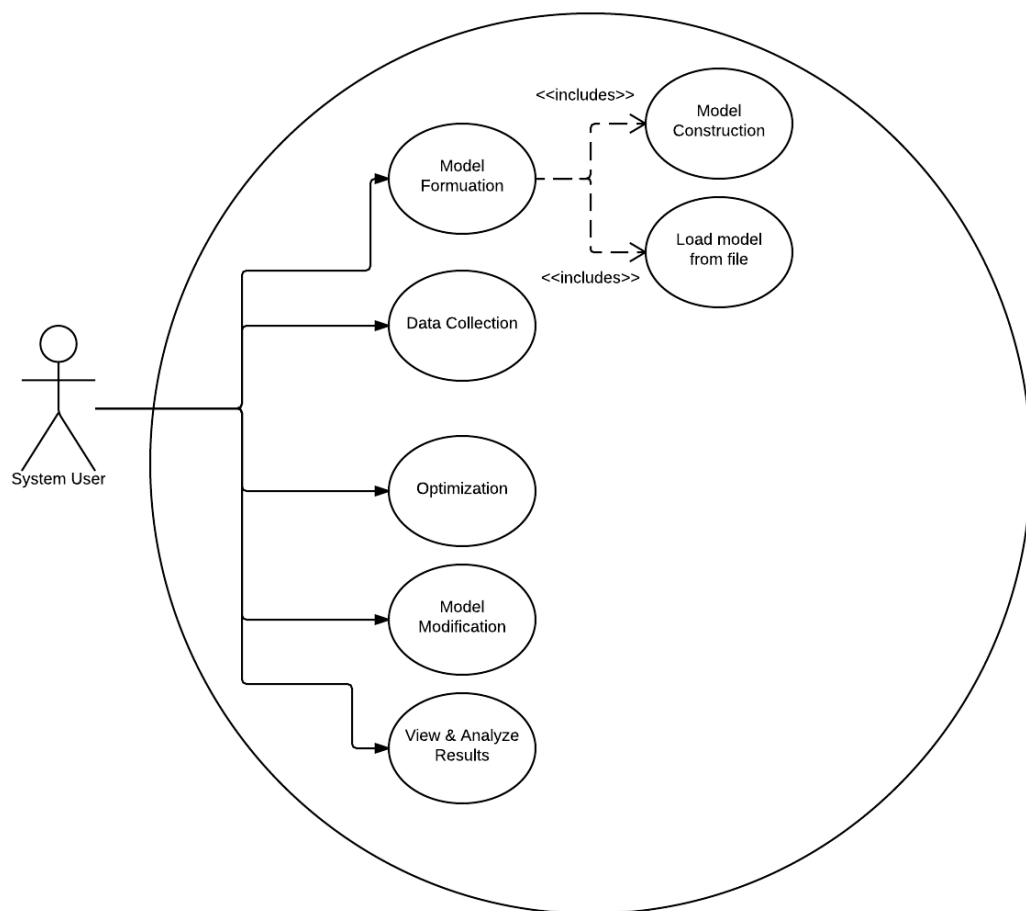


Figure 15 Use Case View of Optimera

## Use Case View of Model Construction

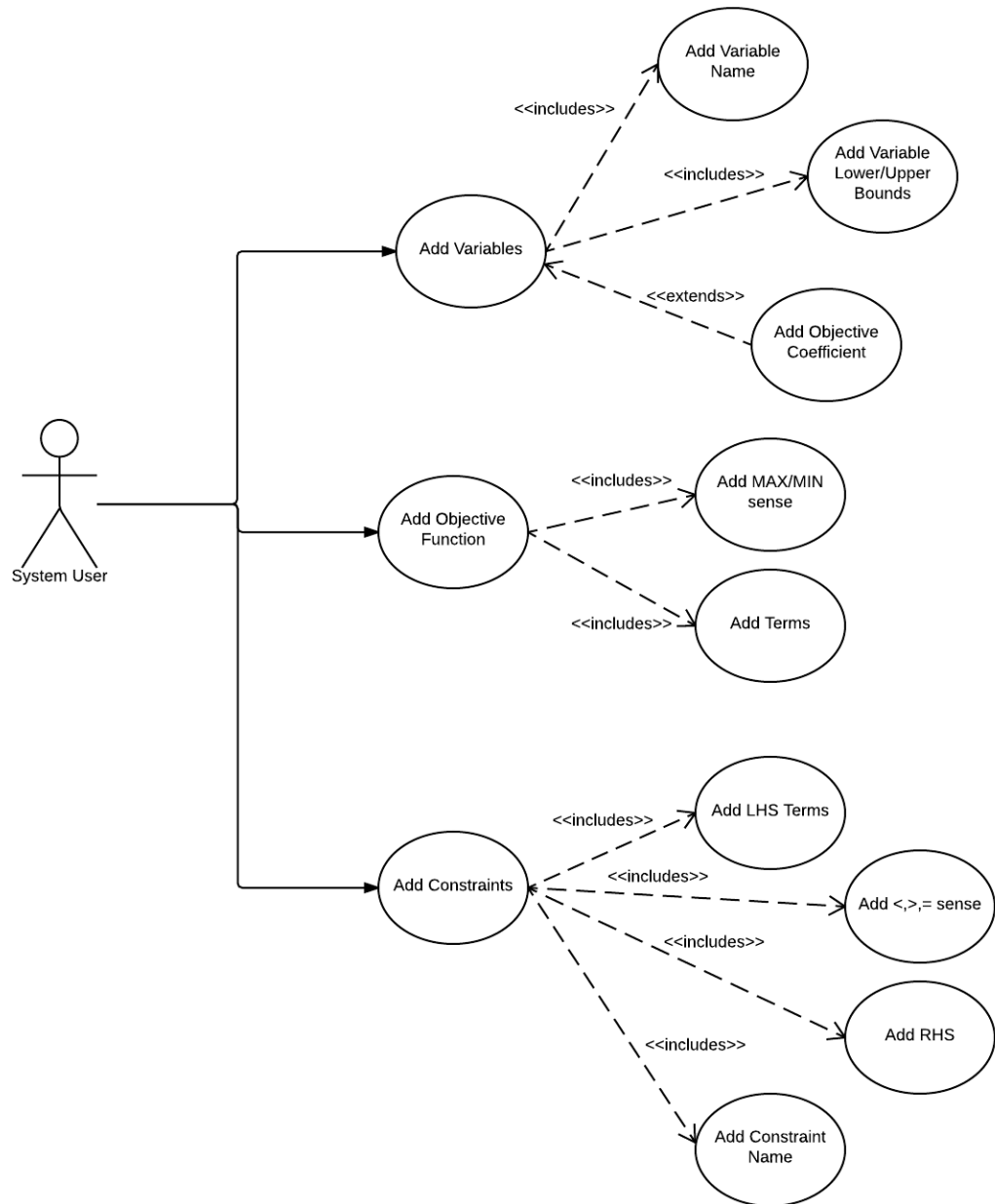


Figure 16 Use Case View of Model Construction

## Use Case View of Model Modification

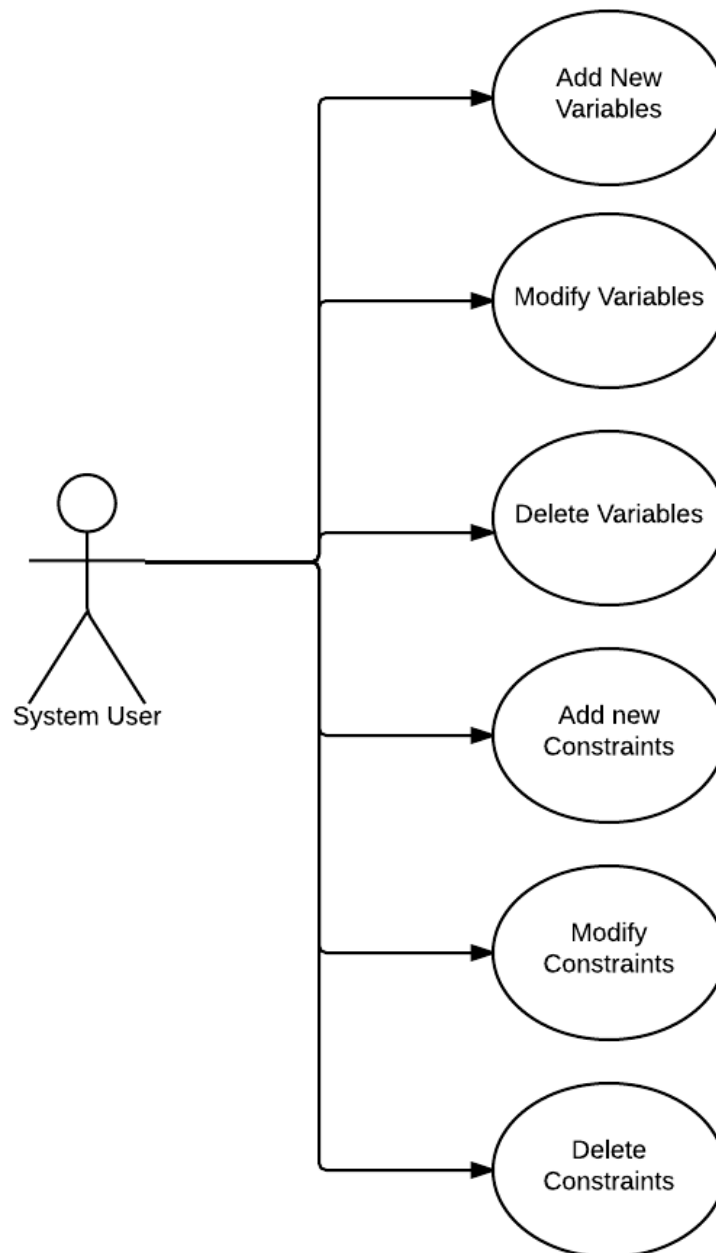


Figure 17 Use Case View of Model modification

### **3.5 System Evaluation Methodology**

The test strategy consists of a series of different tests that will fully exercise the Optimera system. The primary purpose of these tests is to uncover the systems limitations and measure its full capabilities. A list of various planned tests brief explanations follow:

#### **3.5.1 System Test**

The System tests will focus on the behavior of the system that is shown in the use case diagram mentioned in Section 3.4. User scenarios will be executed against the system as well as screen mapping and error message testing. Overall, the system tests will test each of these use cases and the integrated system and verify that it meets the requirements defined in the requirements document.

#### **3.5.2 Stress and Volume Test**

We will subject the system to high input conditions and a high volume of data during the peak times. This will test the system for extreme values of input.

#### **3.5.3 Usability Test**

Optimera is intended to be used by non-technical personnel with little or no experience with the functioning of solvers. Thus Optimera uses windows forms in C# along with advanced graphics libraries to provide easy, aesthetically pleasing, user friendly interfaces that support data input and well defined reporting. The usability will be tested by performing end-user testing. Random people will be asked to use the software and read/build a model.

The usability testing will be supplemented with Documentation testing, Beta Testing and User Acceptance Testing. In documentation Testing, Tests will be conducted to check the accuracy of the user documentation. These tests will ensure that no features are missing, and the contents can be easily understood. The purpose of

these tests is to confirm that the system is developed according to the specified user requirements and is ready for operational use.

# **IMPLEMENTATION AND RESULT DISCUSSION**

This section explains the procedure used to implement the design and some of the challenges faced. The class diagram and code snippets are added for each module for further study of the implementation. Some problems have been modelled and solved to discuss the result of the system developed.

## **4.1. Overall System Implementation**

The system is a Windows Desktop Application. It shall be built in .NET Framework using Gurobi Library [17] for optimization algorithms. As shown in Figure 10 – Class Diagram, each model shall be represented as an object of class GRBModel, consisting of a set of decision variables (objects of class GRBVar), a linear objective function on those variables (specified using GRBModel.SetObjective), and a set of constraints on these variables (objects of class GRBConstr, GRBQConstr, or GRBSOS). Each variable shall have an associated lower bound, upper bound, and type (continuous, binary, etc.). Each linear constraint shall have an associated sense (less-than-or-equal, greater-than-or-equal, or equal), and right-hand side value.

Linear constraints shall be specified by building linear expressions (objects of class GRBLinExpr), and then specifying relationships between these expressions (for example, requiring that one expression be equal to another).

The optimization model could be specified all at once, by loading the model from a file (using the appropriate GRBModel constructor), or built incrementally, by first constructing an empty object of class GRBModel and then subsequently calling GRBModel.AddVar or GRBModel.AddVars to add additional variables, and GRBModel.AddConstr or GRBModel.AddQConstr to add additional constraints.



Models shall be dynamic; that is the user could always add or remove variables or constraints and re-optimize the model.

## 4.2. Module Implementation (Interfaces)

### 4.2.1 Model Construction

The interface prompts the user to build a model from scratch by creating variables and constraints, and adding them to an optimization model using lazy updates. The use case below shows the steps involved in model construction.

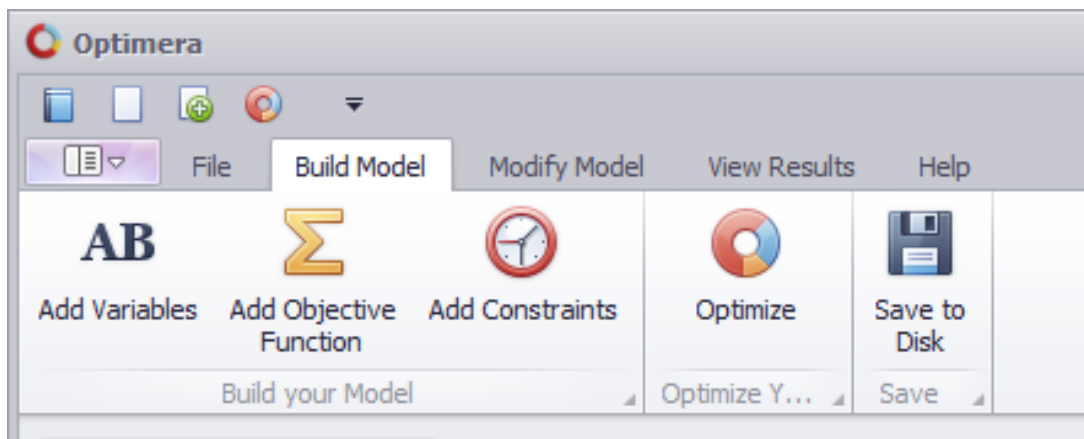


Figure 18 Menu Options to Construct Model

Step 1: The user is prompted to name the model.

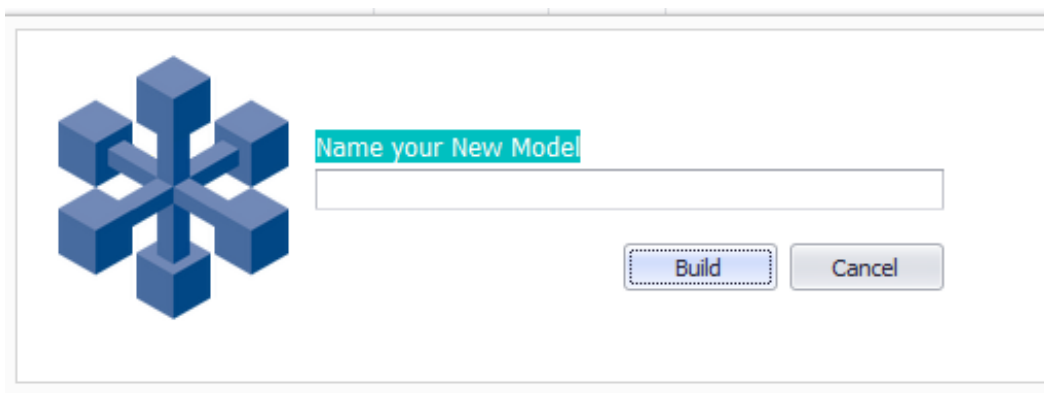


Figure 19 the user is prompted to name the model.



Step 2: The user adds variable by providing a variable name, type, objective coefficient, lower bound and upper bound.

Formulating Model - Step 1: Add Variables

Add Variables

Variable Name: y

Variable Type: BINARY

Objective Coefficient: CONTINUOUS

Variable LowerBound: INTERGER

Variable UpperBound: SEMI-CONTINUOUS

SEMI-INTERGER

☐ Infinite

Add Variable

Your Variables:

0 > x > 1

Cancel OK Next >

Figure 20 the user adds variable by providing a variable name, type, objective coefficient, lower bound and upper bound.

Step 3: The user adds objective function by listing down the left side linear expression consisting of variables, and sense.

Building Model - Step 2: Add Objective Function

**Add Sense**

Do you want to maximize or minimize your objective function?

Minimize

**Add Term(s) to your objective function**

Coefficient: 1

Variable Name: [Dropdown menu]

Undo Add Term

**Objective Function**

Minimize

[Large text area for linear expression]

Clear Add Objective Function

< Back Cancel OK Next >

Figure 21 the user adds objective function by listing down the left side linear expression consisting of variables, and sense.

Step 4: The user adds constraint by listing down the left side linear expression consisting of variables, the constraint sense and the right hand side.

Building Model - Step 3: Adding Constraints

Build the variable side of the constraint equality

Coefficient:

Variable Name:

Add Term Undo

Add Comparison

Comparator:

Add number at the Right Hand Side

RHS:

Add RHS

Name your Constraint

Building Constraint:

Clear

Name

Add

<Back Cancel OK Optimize

Figure 22 the user adds constraint by listing down the left side linear expression consisting of variables, the constraint sense and the right hand side.

These steps ensure that a model is constructed.

## 4.2.2 Model Modification

The user can modify the variables, constraint and objective function of the model already constructed.

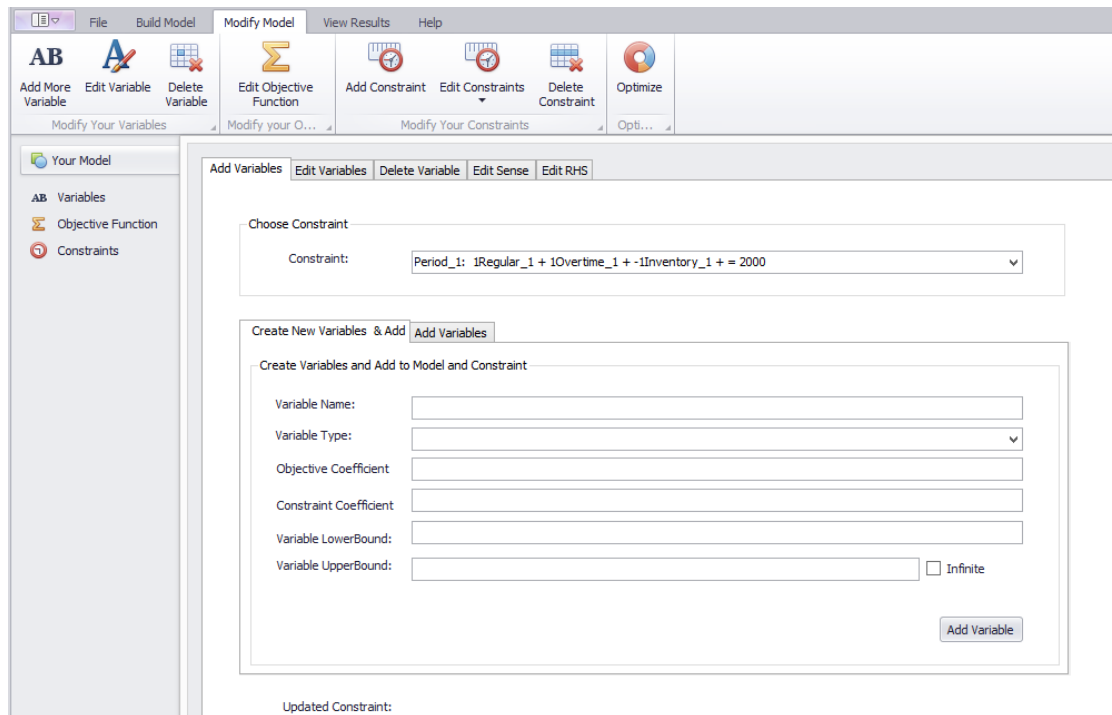


Figure 23 the user can modify the variables, constraint and objective function of the model already constructed.

### 4.2.3 Save Model to Disk

The user can save the constructed or modified model to disk.

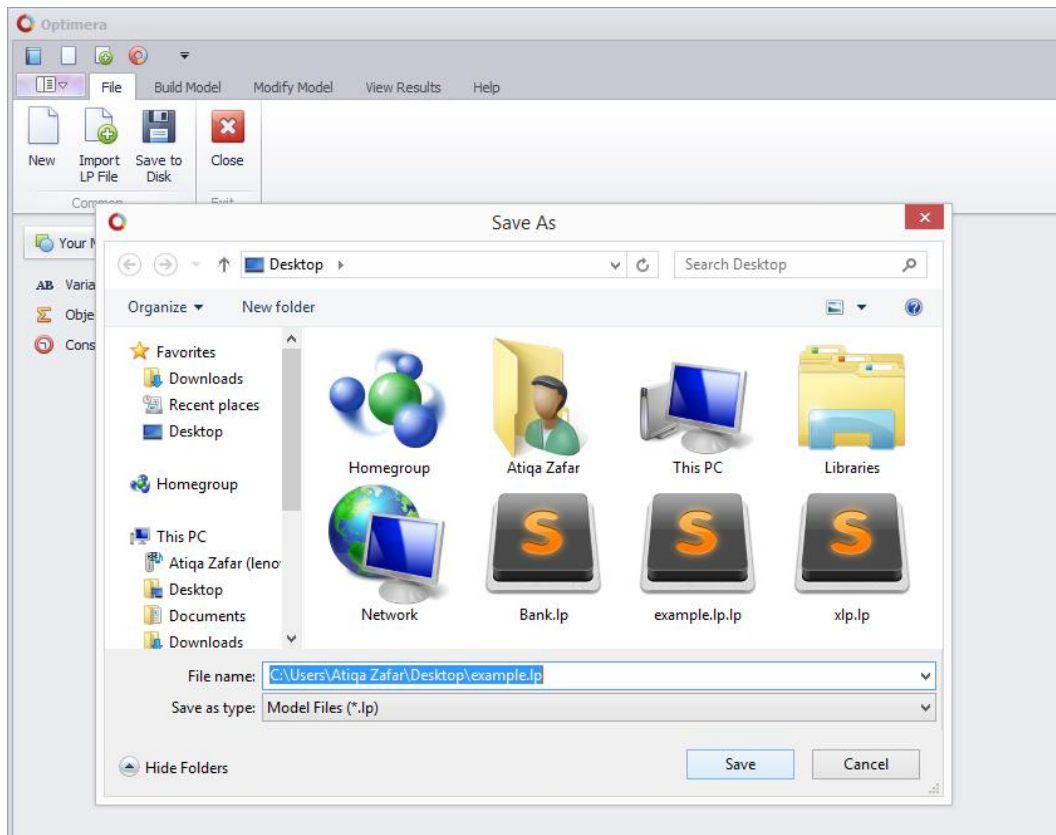


Figure 24 the user can save the constructed or modified model to disk.

#### 4.2.4 Import Model from Disk

The user can import a model file of .lp file format to load a model from disk. This model can be easily modified using the model modification options available.

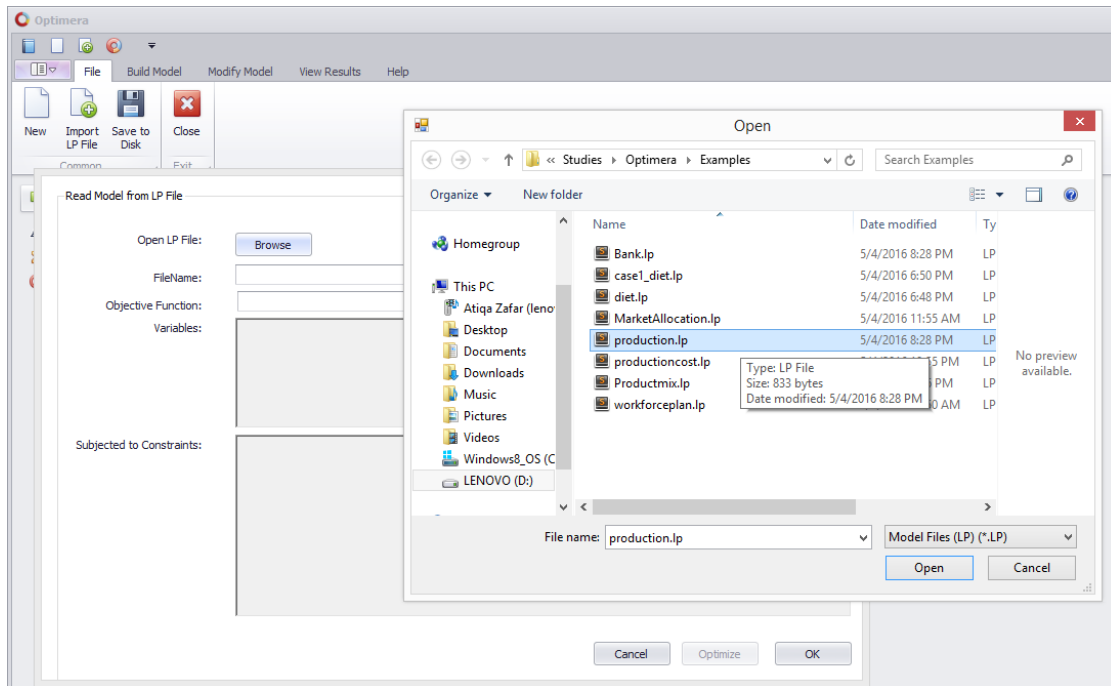


Figure 25 the user can import a model file of .lp file format



## 4.2.5 Optimize

The user can optimize a model constructed or loaded into the software. If an optimal solution is found, the values for decision variables are displayed in a worksheet that can be filtered and sorted.

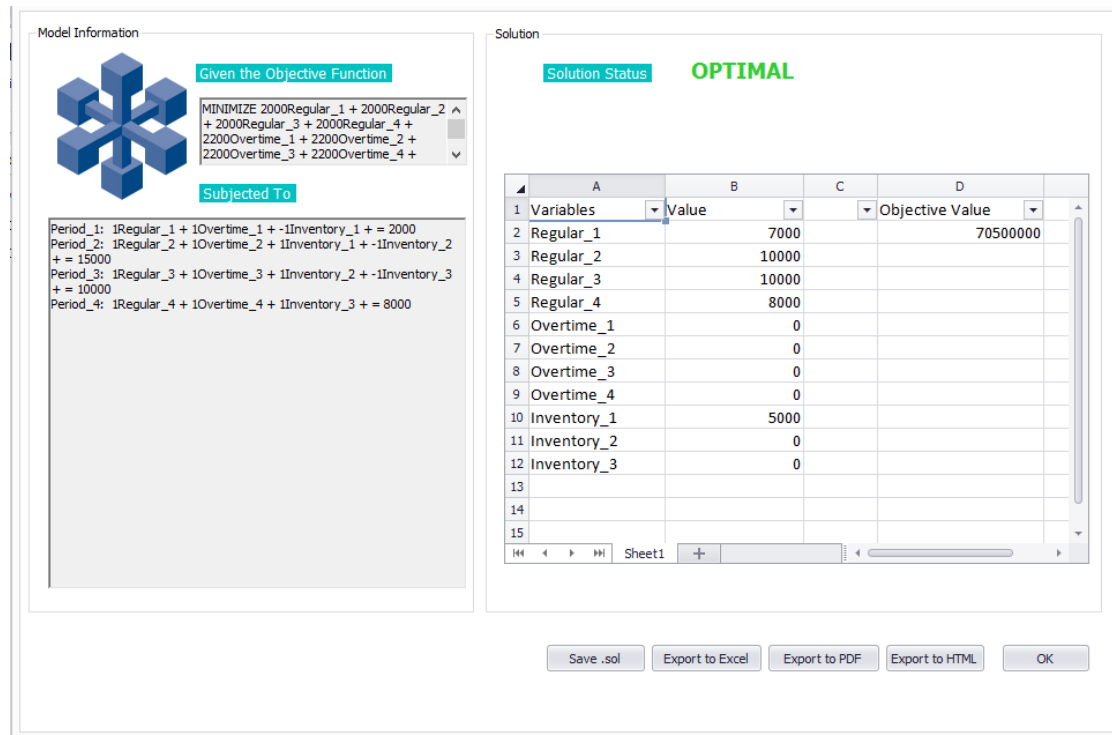


Figure 26 optimal result found

If the solution is found to be infeasible, that is if no solution exists which satisfies all the constraints, the software computes the Irreducible Infeasible Subset (IIS) of the model. An IIS is a minimal set of constraints and variable bounds which is infeasible, but becomes feasible if any constraint or bound in it is removed. The constraints in IIS are displayed and the user is prompted use IIS iteratively to find and remove all conflicting constraints until the problem cannot be solved. Note that there may be additional conflicts besides what is reported via IIS.

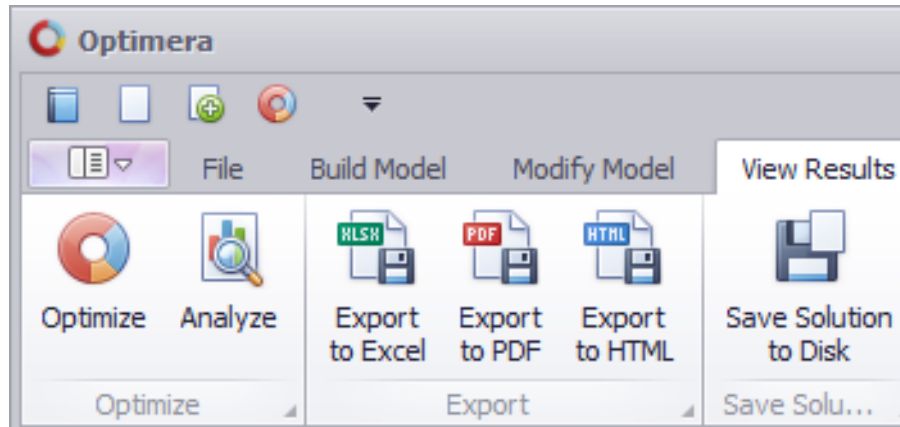


Figure 27 Infeasible model

The model can also become unbounded, that is if the objective function may be improved indefinitely without violating the constraints and bounds. An unbounded model is not processed further and the user is notified.

#### 4.2.6 Export results to .sol, .xlsx, .pdf and .html

The user can export the solution found from optimization to .sol file format, Excel



spreadsheet, PDF and HTML. All the options are provided in the menu ribbon.

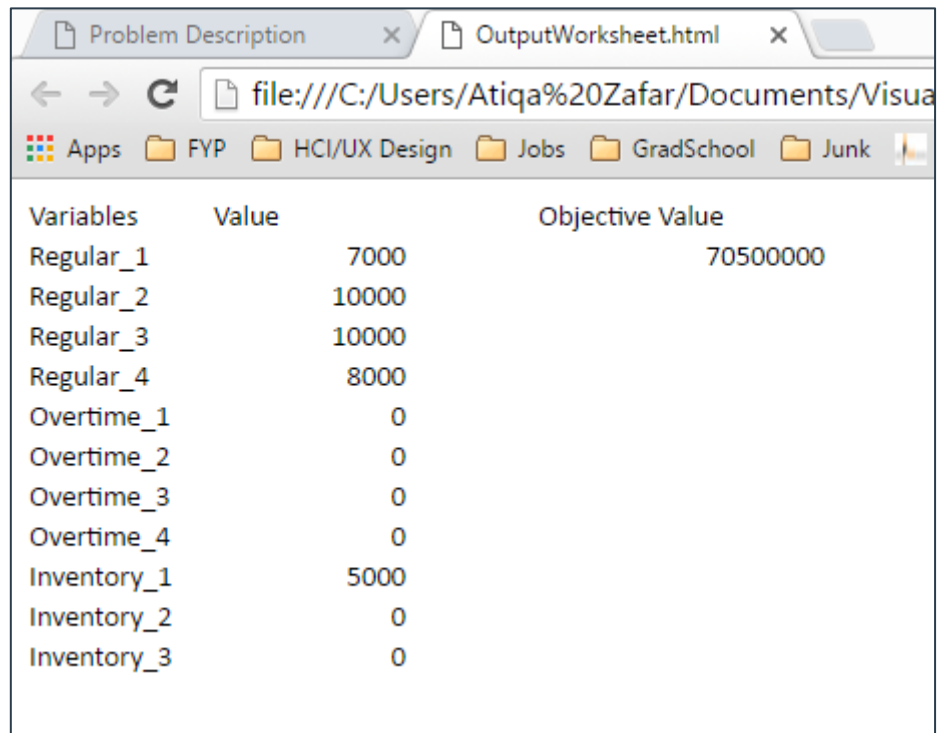
Figure 28  
Menu options for Solution Export

	A	B	C	D	E	F
1	Variables	Value		Objective Value		
2	Regular_1	7000		70500000		
3	Regular_2	10000				
4	Regular_3	10000				
5	Regular_4	8000				
6	Overtime_1	0				
7	Overtime_2	0				
8	Overtime_3	0				
9	Overtime_4	0				
10	Inventory_1	5000				
11	Inventory_2	0				
12	Inventory_3	0				
13						
14						
15						
16						
17						
18						
19						
20						
21						
22						
23						

Figure 29 Solution exported to .XLSX

Variables	Value	Objective Value
Regular_1	7000	70500000
Regular_2	10000	
Regular_3	10000	
Regular_4	8000	
Overtime_1	0	
Overtime_2	0	
Overtime_3	0	
Overtime_4	0	
Inventory_1	5000	
Inventory_2	0	
Inventory_3	0	

Figure 30 Solution exported to .PDF



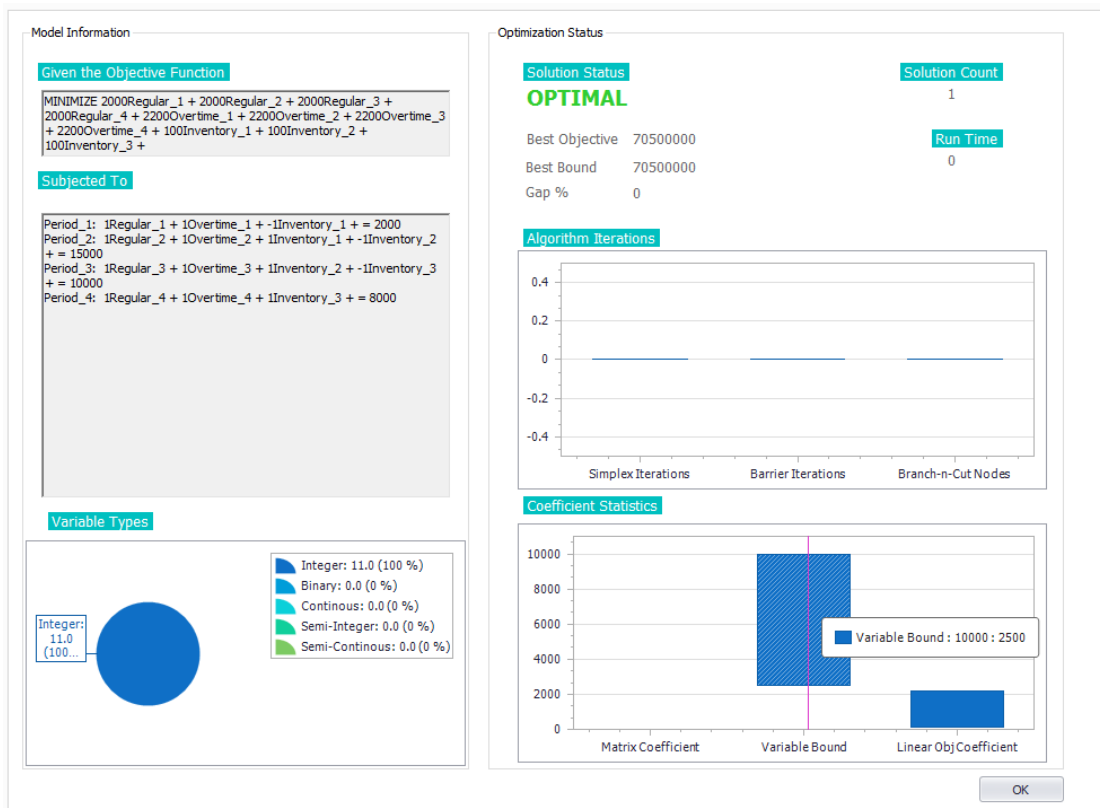
The screenshot shows a web browser window with two tabs: 'Problem Description' and 'OutputWorksheet.html'. The address bar shows the file path: 'file:///C:/Users/Atiqah%20Zafar/Documents/Visua'. The browser's navigation bar includes icons for 'Apps', 'FYP', 'HCI/UX Design', 'Jobs', 'GradSchool', and 'Junk'. The main content area displays the same table as Figure 30, with the 'Objective Value' column showing a value of 70500000 for the first row and subsequent rows.

Variables	Value	Objective Value
Regular_1	7000	70500000
Regular_2	10000	
Regular_3	10000	
Regular_4	8000	
Overtime_1	0	
Overtime_2	0	
Overtime_3	0	
Overtime_4	0	
Inventory_1	5000	
Inventory_2	0	
Inventory_3	0	

Figure 31 Solution exported to .HTML

## 4.2.7 Visualize Results

The user can use the visualization tool to analyze the model and optimization status. The visualization window allows the user to analyze the types of variables involved, solution status, solution count, run time, bounds, algorithmic iterations and coefficient statistics. This tool helps those who are a little more interested in the optimization of their model than just getting an optimal value for the objective function and decision variables.



### 4.3. Module Implementation (Code)

The code snippets for each module have been stored in the Compact Disks provided with this document. Each source code file is named after the module and is properly commented for ease of future developers.

### 4.4. System Evaluation & Result Discussion

Optimera is intended to be used by non-technical personnel with little or no experience with the functioning of solvers. Thus Optimera uses windows forms in C# along with advanced graphics libraries to provide easy, aesthetically pleasing, user friendly interfaces that support data input and well defined reporting.

The usability has been tested by performing end-user testing. Students were asked to use the software and read/build a model by entering extreme data values or null values as input, by entering incorrect data values as input, and by entering correct data. In some cases, the users were given no help or guidance about the interface buttons and were asked to perform model modification, import, optimization, modification and export.

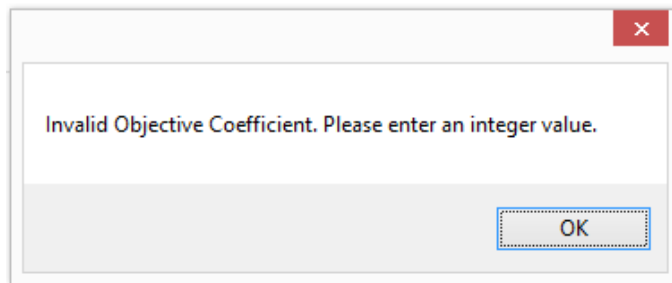


Figure 32 Error message displayed when user entered text into integer input field

The usability has been ensured by providing validation check on all input values. Any value out of bound is immediately notified by an error message to the user and corrected.

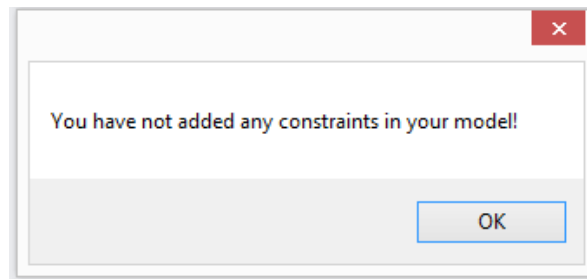


Figure 33 Error message displayed when user tries to edit constraint when the model has no constraints

The figures show screenshots of the few mistakes user made entering the data and how the validation checks ensured proper input.

The System has also been evaluated for efficiency, performance and scalability requirements by modelling and solving model examples based on Case studies from Quantitative Methods for the Management Sciences 45-760 CMU, Pittsburgh, USA Fall 1998 [4] and Gurobi linear programming problems [16].

### CONCLUSION AND FUTURE RECOMMENDATIONS

Optimera has been proposed and developed as a .NET solution for constraint optimization for scheduling and resource allocation problems using linear programming solvers. It allows you to perform hands-on interaction and experimentation with optimization models. You can read models from files, perform complete or partial optimization runs on them, change parameters, and modify the models, re-optimize, and so on. You can easily import data from external systems and export your completed results back again giving you the freedom to schedule the way you want.

However, it has certain limitations and provides only groundwork for future developments in the following areas:

- i. **Serviceability:** Currently, Optimera is solving linear programming problems and LP Models. In future, it's interface and modules can be extended to service resource handling problems of multiple types as follows:
  - a. quadratic programming problems (QP) [46]
  - b. quadratically constrained programming problems (QCP)
  - c. mixed-integer quadratically constrained programming problems (MIQCP)
- ii. **Solver options:** Currently, Optimera deploys Gurobi optimization libraries to solve problems. In future, multiple applications can be built on the same principle, each deploying a different solver library or a combination.
- iii. **Specialization:** Using this project as a proof of concept, specialized applications can be built to cater to specific needs of an organization that lacks technical expertise to model and solve using AMPL or other programming languages.



Nevertheless, Optimera is a proof of how we can provide productivity without complexity. It is developed to provide optimal solution for various resource allocation problems. We have discussed the unmet need or problem this FYP is aiming to solve and the potential customers who might be interested in this solution. We surveyed the market for already deployed solutions and saw what's missing. Optimera is different from all the other products in the market for the value it provides. It can be deployed in any setting i.e. academic, business etc and can be used to solve constrained optimization problems very easily and effectively. With Optimera, your problem's optimal solution is just a click away!

## ***REFERENCES***

- [1] Alexander Bockmayr & J. N. Hooker “Constraint programming”  
<http://web.tepper.cmu.edu/jnh/cp-hb.pdf>  
May 2003
- [2] Barták, R: “Constraint Programming: In Pursuit of the Holy Grail”  
<http://kti.mff.cuni.cz/~bartak/downloads/WDS99.pdf>  
Charles University, Faculty of Mathematics and Physics, Department of Theoretical Computer Science, Prague 2002
- [3] Barták, R. “On-line Guide to Constraint Programming”  
<http://kti.mff.cuni.cz/~bartak/constraints/>  
Prague, 1998  
Last accessed: March 2016
- [4] Chow, K.P., Perett, M. “Airport Counter Allocation using Constraint Logic Programming”  
Proc. of Practical Application of Constraint Technology (PACT97), London, UK, 1997
- [5] CLP (COIN-OR LP) <https://projects.coin-or.org/Clp>  
Last accessed: March 2016
- [6] CourseLeaf Section Scheduler (CLSS)  
<http://www.courseleaf.com/section-scheduler/>  
Last accessed: March 2016
- [7] Darby-Dowman, K., J. Little, G. Mitra, M. Zaffalon. “Constraint logic programming and integer programming approaches and their collaboration in solving an assignment scheduling problem”1997.
- [8] De Farias, I. R., E. L. Johnson and G. L. Nemhauser. “A branch-and-cut approach without binary variables to combinatorial optimization problems with continuous variables and combinatorial constraints” Georgia Institute of Technology 1999.
- [9] Event Management Systems (EMS)  
<http://www.dea.com/solutions/software-solutions/Solutions.aspx?mi=1>  
Last accessed: March 2016
- [10] EZnet scheduler  
<https://www.eznetscheduler.com/>  
Last accessed: March 2016

[11] FICO XPRESS <http://www.fico.com/en/products/fico-xpress-optimization-suite>  
Last accessed: March 2016

[12] Focacci, F., Lamma, E., Mello, P., Milano, M  
“Constraint Logic Programming for the Crew Rostering Problem”  
Proc. of Practical Application of Constraint Technology (PACT97), London, UK,  
1997

[13] Francesca Rossi, Peter van Beek, Toby Walsh  
[https://cs.uwaterloo.ca/~vanbeek/Publications/kr\\_handbook06.pdf](https://cs.uwaterloo.ca/~vanbeek/Publications/kr_handbook06.pdf)  
Constraint Programming  
2006

[14] Freuder, E.C. “Synthesizing Constraint Expressions”  
Communications ACM 21(11): 958-966, ACM, 1978

[15] Freuder, E.C., Wallace, R.J. “Partial Constraint Satisfaction”  
Artificial Intelligence, 1-3(58): 21-70, 1992

[16] Gurobi, “Gurobi Optimizer Reference Manual”  
<http://www.gurobi.com/documentation/6.5/refman.pdf>  
Last accessed: March 2016

[17] Gurobi, “Gurobi .NET API Overview”  
[http://www.gurobi.com/documentation/6.5/refman/cs\\_net\\_api\\_overview.html](http://www.gurobi.com/documentation/6.5/refman/cs_net_api_overview.html)  
Last accessed: March 2016

[18] GLPK (GNU Linear Programming Kit) <https://www.gnu.org/software/glpk/>  
Last accessed: March 2016

[19] Hans Mittelmann, “Benchmarks for Optimization Software”  
<http://plato.asu.edu/bench.html>  
Last accessed: March 2016

[20] Helmut Simonis “Building Industrial Applications with Constraint Programming”  
<http://4c.ucc.ie/~hsimonis/ccl2.pdf>  
University College Cork, 2001

[21] Heipcke, S. “Combined Modelling and Problem Solving in Mathematical Programming and Constraint Programming”, Ph.D. Thesis, University of Buckingham. 1999

[22] Helmut Simonis “Models for Global Constraint Applications”

<http://link.springer.com/article/10.1007%2Fs10601-006-9011-7?LI=true#/page-1>  
Springer, 29 November 2006

[23] Hooker, J. N. "Logic-based methods for optimization"  
A. Borning, ed., Principles and Practice of Constraint Programming, Lecture Notes in  
Computer Science 874 336-349 1994.

[24] IBM ILOG CPLEX <http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/>  
Last accessed: March 2016

[25] Igor Grešovnik "Simplex algorithms for nonlinear constraint optimization  
problems" Ljubljana, 2009

[26] Iiro Harjunkskia, Christos T. Maraveliasb, John Wassicki  
"Scope for industrial applications of production scheduling models and solution  
methods"  
<http://www.sciencedirect.com/science/article/pii/S0098135413003682>  
May 2013

[27] Jaffar, J. & Lassez J.L. "Constraint Logic Programming"  
Proc. The ACM Symposium on Principles of Programming Languages, ACM, 1987

[28] Javier Larrosa, Albert Oliveras, Enric RC, "Mixed Integer Linear programming"  
<https://www.cs.upc.edu/~erodri/webpage/rpar/s6.pdf>

[29] J. N. Hooker "Logic, Optimization and Constraint Programming"  
<http://wpweb2.tepper.cmu.edu/jnh/joc2.pdf>  
April 2000; Revised November 2001

[30] Laundry, R. S. "Logically Constrained Mathematical Programming Problems",  
Ph.D. thesis, University of Southampton. 1986

[31] Lawler, E.W., Wood, D.E. "Branch-and-bound methods: a survey, in: Operations  
Research" 1966

[32] LP SOLVE <http://sourceforge.net/projects/lpsolve/>  
Last accessed: March 2016

[33] Marriot, K. & Stuckey. P. "Programming with Constraints: An Introduction"  
<https://mitpress.mit.edu/books/programming-constraints>  
The MIT Press, Cambridge, Mass., 1998

[34] Michael T. & Gerald C., “Quantitative Methods for the Management Sciences”  
Carnegie Mellon University, Pittsburgh, PA 15213 USA  
Fall 1998

[35] Michela Milano,  
“Constraint Programming Approach to AI Applications”  
<http://cse.unl.edu/~choueiry/F02-421-821/Documents/Milano.pdf>  
Tutorial 3: AI\*IA99 Bologna September 99

[36] Mimoso Scheduling Software  
<http://www.mimosasoftware.com/>  
Last accessed: March 2016

[37] Robert Fourer “Using AMPL with Gurobi to Apply Optimization Models  
Efficiently and Reliably”  
[http://ampl.com/MEETINGS/TALKS/2011\\_12\\_Tokyo\\_October\\_Sky\\_Symp.pdf](http://ampl.com/MEETINGS/TALKS/2011_12_Tokyo_October_Sky_Symp.pdf)  
Gurobi Optimizer Solution Seminar Tokyo, 2 December 2011

[38] Sara Velez, Christos T. Maravelias  
“A branch-and-bound algorithm for the solution of chemical production scheduling  
MIP models using parallel computing”  
<http://www.sciencedirect.com/science/article/pii/S0098135413000951>  
December 2012

[39] SCIP <http://scip.zib.de/>  
Last accessed: March 2016

[40] Shuonan Dong “Methods for Constrained Optimization”  
<https://www.fd.cvut.cz/personal/nagyivan/erasmus/constrainedoptimization.pdf>  
Spring 2006’

[41] SoPlex <http://soplex.zib.de/>  
Last accessed: March 2016

[42] T. Fröhlichwirth & S. Abdennadher. “Essentials of Constraint Programming.”  
Springer, 2003.

[43] Time Tabler  
<http://www.timetabler.com/>  
Last accessed: March 2016

[44] University of Texas at Austin, “Algorithms for Constrained Optimization”  
[https://www.me.utexas.edu/~jensen/ORMM/supplements/units/nlp\\_methods/const\\_opt.pdf](https://www.me.utexas.edu/~jensen/ORMM/supplements/units/nlp_methods/const_opt.pdf)

[45] Van Hentenryck, P. “Constraint Satisfaction in Logic Programming”  
The MIT Press, Cambridge, Mass., 1989

[46] Wikipedia, the free encyclopedia “Quadratic Programming”  
[https://en.wikipedia.org/wiki/Quadratic\\_programming](https://en.wikipedia.org/wiki/Quadratic_programming)  
Last accessed: March 2016

[47] Wikipedia, “Constraint Optimization Wikipedia”  
[https://en.wikipedia.org/wiki/Constrained\\_optimization](https://en.wikipedia.org/wiki/Constrained_optimization)  
Last accessed: March 2016

[48] Wikipedia, “Integer Programming”  
[https://en.wikipedia.org/wiki/Integer\\_programming](https://en.wikipedia.org/wiki/Integer_programming)  
Last accessed: March 2016