Georgia State University
ScholarWorks @ Georgia State University

Mathematics Theses | Department of Mathematics and Statistics

8-7-2007

An Optimal Solution on Screening and Treatment of Chlamydia Trachomatis and Neisseria Gonorrhoeae

Xin Wei

Follow this and additional works at: http://scholarworks.gsu.edu/math_theses

Recommended Citation
Wei, Xin, "An Optimal Solution on Screening and Treatment of Chlamydia Trachomatis and Neisseria Gonorrhoeae." Thesis, Georgia State University, 2007.
http://scholarworks.gsu.edu/math_theses/35

This Thesis is brought to you for free and open access by the Department of Mathematics and Statistics at ScholarWorks @ Georgia State University. It has been accepted for inclusion in Mathematics Theses by an authorized administrator of ScholarWorks @ Georgia State University. For more information, please contact scholarworks@gsu.edu.

AN OPTIMAL SOLUTION ON SCREENING AND TREATMENT OF CHLAMYDIA TRACHOMATIS AND NEISSERIA GONORRHOEAE

by

XIN WEI

Under the Direction of Dr.Guantao Chen

ABSTRACT

We propose a resource allocation model for the management of the fund for the screening and treatment of women infected by *Chlamydia trachomatis* and *Neisseria gonorrhoeae* . The goal is to maximize the number of infected women cured of *Chlamydia trachomatis* and *Neisseria gonorrhoeae* infections. The population going for screening is divided into groups by ages and races. The group number is dynamic. Different groups have different infection rates. There are four possible test assays and four possible treatments. We employed a two-phase algorithm to solve the problem. The first phase is small so an exhaustive method is applied, while the second phase is transformed to a knapsack problem and a dynamic programming method is applied.

INDEX WORDS: operations research, mix-integer programming, dynamic programming, knapsack problem

AN OPTIMAL SOLUTION ON SCREENING AND TREATMENT OF CHLAMYDIA TRACHOMATIS AND NEISSERIA GONORRHOEAE

by

XIN WEI

A Thesis Submitted in Partial Fulfillment of the Requirement for the Degree Of

Master of Science

in the college of Arts & Sciences

Goergia State University

2007

Copyright by

Xin Wei

2007

AN OPTIMAL SOLUTION ON SCREENING AND TREATMENT OF CHLAMYDIA TRACHOMATIS AND NEISSERIA GONORRHOEAE

by

XIN WEI

Major Professor: Dr.Guantao Chen

Committee: Dr. Guantao Chen
Dr. Guoyu Tao
Dr. George Davis

Electronic Version Approved:
Office of Graduate Studies
College of Arts and Sciences
Georgia State University
August, 2007

ACKNOWLEDGMENTS

This thesis would not have been possible without the assistance and support from many people who gave their supports in different ways. To them I would like to express my deepest gratitude and sincere appreciation.

First of all, I would like to thank my advisor Dr. Guantao Chen for all his encouragement and insightful guidance. Without his giving me this opportunity to do this project, this thesis would never come out. I would never get so interesting and intensive practice and exercise on Math and Programming.

I would also appreciate Dr. Guoyu Tao for giving me such an invaluable opportunity to do this project as well as his suggestions and critical review of the draft.

Thanks to all the people who directed and helped me before, they are Josh Liu, Kun Zhao, Fasheng Qiu and Hai Deng. Without their help, I could by no means acheive the goal of this research.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS iv

LIST OF TABLES vii

Chapter

1 Introduction – Preliminaries 1

1.1 Background of the Problem 2

2 Linear Programming 5

2.1 The Linear Programming Model 6

2.2 Assumptions of Linear Programming 10

2.3 Integer Programming 11

2.4 Binary Integer Programming 12

3 Models 15

3.1 Description of the model 15

3.2 Data used in the model 17

3.3 The mathematical model 21

3.4 The Algorithm 27

4 Results and Discussions 31

4.1 Results 31

4.2 Future Work 34

BIBLIOGRAPHY 36

APPENDIX 38

LIST OF TABLES

Table Page

3.1 Variables used in the optimal resource allocation model 18

4.1 Optimal Strategy 31

4.2 population group 33

CHAPTER 1

INTRODUCTION – PRELIMINARIES

Chlamydia trachomatis (CT) and Neisseria gonorrhoeae (GC) rank as the two most commonly reported sexually transmitted diseases (STDs) in the United States [1]. In 2002, more than 834,500 cases of CT were reported to the Centers for Disease Control and Prevention (CDC) out of an estimated 3 million cases [2] . The number of GC infections reported for the same year was over 351,800. Untreated CT and GC infections may result in pelvic inflammatory disease (PID), ectopic pregnancy and infertility, with the healthcare cost of untreated CT and GC infections and their attributable PID estimated at over $2 billion annually [3, 4].

The CDC through its Division of Sexually Transmitted Disease(DSTD), and in coolaborative efforts with health-care providers has been continually working to evolve effective ways to manange and control CT and GC infections. At the Health Services Research and Evaluation Branch(HSREB) of DSTD, there is ongoing research to find effective ways to deploy the limited resources at the disposal of family planning and DSTD clinics to combat the spread of CT and GC. The preventive Health Amendments of 1992 provided $8.3 million in fiscal year 1994 for the prevention of infertility associated primarily with CT. As of December 2000, all 50 states in the US and District of Columbia had enacted laws requiring the reporting of CT infections. CT-related activities authorized by legislation include detection, treatment, follow-up, and referral services for at-risk and their sex-partners. In 1994, the direct anuual cost resulting form CT cases and its attributable PID was estimated to be $ 2 billion by one study. Moreover, up to 70% of CT infections and 50% of GC infections are asymptomatic [5, 6, 7, 8].

1.1 Background of the Problem

Although annual screning for CT and GC is recommended for sexually active adolescents and young women, many publicly funded clinic programs may not have sufficient budgets to screen all eligible women with the most effective tests and to offer all eligible women a more expensive, single-dose treatment that optimizes compliance. As a result, the best strategy to detect and treat the most infections is through universal screening of at-risk populations. However, most public health programs or clinics may not have the resources to screen all patients with the most effective test and treatment. Thus, to effectively use limited resources, CT and GC control programs usually provide selective screening based on some defined guidelines. The CDC recommends that all sexually active women below the age of 25 year be screened at least once annually for CT [9]. Unlike in the case of CT, there is no established recommendation to screen for GC, see U.S. Preventive Services Task Force(USPSTF) recommendations: there is a recommendation for at risk women. The highly non-uniform demographical distribution of the infection may be among the reasons why no such recommendation exists. In the absence of established GC screening recommendations for the general population, control programs have used several approaches, including universal screening, selective screening, routine dual treatment, and reflex testing, in which CT positive patients are tested for GC.

Recent studies have reported that up to 45% of family planning and STD clinic female patients that are infected with GC are also infected with CT [10]. Such high coinfection rates may seem to justify the CDC's longstanding recommendation that presumptive treatment for CT be given to women who test positive for GC in populations where the coinfection of rate is $> 20\%$ [11].

To best use their limited resources, many clinics have conducted studies to design ef-

fective screening criteria for their CT/GC control programs. However, for CT/GC control programs, identifying which subpopulations to screen for chlamydial and/or gonorrheal infections is just one part of a complicated problem. The availability of several testing technologies with varying performances and costs present a challenge for program managers when making decisions about selecting test assays. Newer diagnostic tests that are less intrusive and more sensitive offer increased opportunities for screening, but at a greater cost. One question that many clinic managers face is, whether it is better to use a more sensitive and expensive test to screen fewer patients, or rather use a relatively cheaper and less sensitive test to screen a greater number of patients. To further complicate the situation, test manufacturers market combination tests or bundled tests at prices that are more lucrative than the price of a single-pathogen test. This situation encourages the testing for GC even when its prevalence in the population is extremely low.

Many studies have examined the cost-effectiveness of screening for CT and/or GC, as well as the comparative cost-effectiveness of using different testing technologies. However, because the parameters that influence the screening outcomes are many and may vary considerably, a resource allocation model can be used improve the outcomes for each particular setting [12].

Because CT infection is a strong predictor of GC infection and the control programs and resources for the two infections are often integrated, we expanded Tao et al's resource allocation model to include gonorrhoeae. Our objective was to develop a model that found the combination of screening coverage, testing technology and treatment that would maximize the number of CT and/or GC infections cured under a fixed program budget. We also wanted to analyze the threshold prevalence at which screening for CT or GC is cost effective.

This thesis presents a binary linear programming model that determines the optimal

combination of screening and test selection, and treatment for controlling CT and GC infections in asymptomatic women for a fixed budget program. The next chapter discusses binary programming methods used in decision making problem formulations. In Chapter 3, the models to solve the problem are developed based upon related work done by Tao et al, and Chapter 4 discusses the results obtained from the model implementation.

CHAPTER 2

LINEAR PROGRAMMING

The flow of resources in a production process or any quantitative process involves complex inter-relationships among numerous activities. Although differences may exist between the processes involved, as well as between the goals to be achieved, in many cases there are major similarities in the operations of seemingly very different systems. In order to analyze such systems, the working parts (such as capital, raw materials, labor, etc.) of the system have to be first identified, and the goal or objective to be achieved (for example minimization of cost or maximization of profit) has to be set. A mathematical model may then be found for the system. A computational scheme is then developed to determine the best schedule of action among several alternatives to achieve the objective. The designing of such schemes is called mathematical modeling. If a system to be analyzed can be represented by (or approximated to) a model consisting of linear inequalities and its objective also expressible as a minimization or maximization of a linear expression, the analysis of the system is called linear programming.

Linear programming (LP), as a mathematical and operations research technique, is widely used in making quantitative decisions in the management and administration of military, governmental, commercial and industrial systems. Linear programming has been used in several large-scale problems, often resulting in important interventions and huge cost-savings.

2.1 The Linear Programming Model

Generally, a linear programming application involves the problem of finding a combination of different types of activities that efficiently exploits some limited available resources. The solution to a linear programming problem reduces to finding an optimal value (which may be a minimum or a maximum, depending on the problem) of a linear function of several variables (called the objective function) subject to a set of constraints. Let x_j(for $j = 1, 2, ..., n$) be level of activity j and $c_j$be the increase in the *objective function* $Z = f(x)$ that would result from a unit increase in level of activity$j$. Let $b_i$be the amount of resource i (for $i = 1, 2, ..., m$) available for allocation to all activities, and a_{ij} be the amount of resource i consumed by each unit of activity j. A linear programming model may then be formulated as

Maximize

$$Z = c_1x_1 + c_2x_2 + ... + c_nx_n,$$

Subject to

$$a_{11}x_l + a_{12}x_2 + ... + a_{ln}x_n < b_1$$

$$a_{2l}x_1 + a_{22}x_2 + ... + a_{2n}x_n < b_2$$

$$\vdots$$

$$a_{ml}x_1 + a_{m2}x_2 + ... + a_{mn}x_n < b_m$$

and

$$x_1 \geqslant 0, x_2 \geqslant 0, \ldots, x_n \geqslant 0$$

The levels of activities x_i, are called the *decision variables*. The restrictions on the decisions variables, expressed as inequalities in the 'Subject to' part of the formulation are

called constraints. The last set of inequalities, $x_j \geq 0$ are called *nonnegativity constraints*.

Other Forms of LP Formulation

The general LP model we have discussed may be written compactly as

$$\text{maximize}\{Z = cx | Ax \leq b, x \geq 0\},$$

where A is a matrix, b and x are column vectors, and c is a row vector. However, there are many other problems for which their LP formulations are different from the general model described. Other possible forms of LP problem formulations are obtained by making the following modifications:

(1) Minimizing rather than maximizing the objective function:

$$\text{minimize}\, Z = \sum_{j=1}^{n} c_j x_j$$

(2)Functional constraints with $\geq$ inequality:

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n \geq b_i \text{ for some values of } i.$$

(3) Functional constraints in equation form:

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n = b_i \text{ for some values of } i.$$

(4) Making some of the decision variables unrestricted:

$$x_j \text{ unrestricted in sign for some values of } j.$$

Any LP problem that mixes some of the above forms can be reduced to the standard form we described earlier by some algebraic manipulations. For example, a minimization problem may be transformed into a maximization problem by multiplying the objective function by -1.

Solving LP Problems

According to Encyclopedia Britannica, applications of the method of linear programming were first seriously attempted in the late 1930's by Leonid Kantorovich and Wassily Leontiefin the areas of manufacturing schedules. However, little recognition was given to their work at that time. It was after the World War Π, during which linear programming was used extensively to deal with the problem of transportation scheduling and allocation of resources under many constraints that the subject gained acceptability. Then, in 1947 George Dantzig developed the simplex method, which greatly simplified the solution of linear programming problems, giving further impetus to the subject.

Since the 1950s, linear programming has experienced a remarkable growth and is used in many applications, which would have been thought to be too complex and enormous to deal with. It is in this same period that the computer industry has made extraordinary developments. The link between the developments made in operations research (and for that matter, linear programming) and the advances made in the computer industry cannot be over emphasized. The development of high-speed microprocessors and data-processing techniques has brought the breakthrough that operations research needed. It must be noted, however, that the computational resources needed to solve a linear programming problem can be very high and so many problems may still be computationally infeasible.

Two families of solution techniques are widely used in many of today's linear programming software. Both techniques visit a progressively improving series of trial solutions, until a solution is reached that satisfies the conditions for an optimum. The simplex method approach, developed in 1947 by George Dantzig, visits "basic" solutions computed by fixing enough of the variables at their bounds to reduce the constraints $Ax = b$ to a square system, which can be solved for unique values of the remaining variables. Basic solutions represent

extreme boundary points of the feasible region defined by $Ax = b, x > 0$, and the simplex method can be viewed as moving from one such point to another along the edges of the boundary. Barrier or interior-point methods, by contrast, visit points within the interior of the feasible region. The focus of this thesis is to formulate binary programming models for the problem introduced in chapter one, and therefore the details of these solution methods will not be discussed.

There are several computer packages available today that incorporate different variants of the simplex method and interior-point algorithms to solve linear programming problems. These computer packages may be put into two groups: solvers and modeling languages. Solvers are essentially computer implementations of one or more of the available linear programming algorithms. Several vendors market multi-purpose solvers that can be applied to a wide array of linear programming and other optimization problems. Also, specialized solvers may also be developed for particular problems, and may not even be applicable to other similar problems. Some of the most popular solvers available today include CPLEX, FortMP, LDSfDO (Linear INteractive Discrete Optimizer), LPSolve, Solver DLL, and XA, Xpress [16].

Large-scale linear programming models can contain hundreds of constraints, and these constraints are often expressed with indexes and summation notation. Before a solver can be called to solve the problem, these many constraints must be read individually. Modeling languages provide the convenient environment for formulating linear programming models in an easy and efficient way, allowing the use of several arithmetic operators, vectors and summations. They offer several features for model management, data import and export, report generation, and translation of the user-entered model into algorithmic code usable by an available solver. Modeling languages require at 1east one solver, and many vendors offer compatibility with several solvers. Examples of modeling languages include, AIMMS,

AMPL, GAMS, MPL, and LINGO [16].

2.2 Assumptions of Linear Programming

Representing real-world problems by mathematical models often require the imposition of several assumptions since mathematical models often describe ideal, unattainable situations. Therefore, to apply a linear programming formulation to a problem, the fundamental assumptions of proportionality, additivity, divisibility, and certainty have to been met. These assumptions are at the core of the linear programming formulation itself, and must be satisfied together with other assumptions that may be made for a particular problem.

Proportionality is an assumption that the contribution of each activity to the objective function Z is proportional to the level of the activity x_j. Also, the contribution of each activity in the functional constraint equation is proportional to the level of activity the x_j. As a result of this proportionality assumption, the objective function and the functional constraints are all first-degree equations in the variable X_j.

The additivity assumption requires that every function in the linear programming model is the sum of individual contributions of the respective activities. Therefore, cross-product terms (involving the product of two or more variables) are excluded by the additivity assumption. The additivity, together with the proportionality assumption, ensures that all functions in the model are linear combinations of the decision variables.

The divisibility assumption allows the decision variables to be continuous. They can take on integer values as well as fractional values. This assumption, as will be discussed later, may be deliberately violated in certain situations.

Lastly, the certainty assumption allows us to assume that all the parameters of model,

namely, the coefficients of the objective function, the coefficients of the functional constraints, and the amount of available resources, are known constants. In real-world problems, however, model parameters are not always precisely known and maybe subject to other uncontrollable factors.

Due in part to the certainty assumption, it is a well-acknowledged fact that the exact solution obtained from one implementation of an optimization model is not conclusive. After an optimal solution is found, sensitivity analysis is conducted to identify the parameters that influence the optimal solution. By performing sensitivity analysis, we are able to determine ranges for parameters for which the optimal solution stays the same.

2.3 Integer Programming

The divisibility assumption allows solutions to the LP problem to take non-integer values. In practice, however, there are many problems for which the solution makes sense only if the decision variables have integer values. Typical examples involve locating production facilities at different sites, and assigning individuals to operate equipment. If a 1inear programming problem is further constrained for some or all of its decision variables to be integer values, the resulting formulation is called *integer linear programming*, (commonly referred to as *integer programming*). When some, but not all of the decision variables are restricted to integers, the problem is called *mixed integer programming*(MIP).In the case where all the decision variables are restricted to integers, the formulation is called *a pure integer programming.*

Linear and integer programming have proved valuable for modeling many and diverse types of problems in planning, routing, scheduling, assignment and design. Industries that make use of linear programming and its extensions include transportation, energy, telecommunications, and manufacturing of many kinds.

2.4 Binary Integer Programming

When decision variables of an integer programming problem are restricted to just two integer values, 0 or 1, the problem becomes a *binary integer programming* problem. The use of binary integers allows many special formulation techniques to manipulate both the decision variables and the constraint equations. Integer programming problems such as the fixed-charge network flow problem and the famous traveling salesman problem are often expressed in terms of binary variables. The fixed-charge network problem modifies the minimum-cost network flow paradigm by adding a term to the cost, where the binary variable is set to 1 if an arc carries a nonzero flow; it is set to zero otherwise.

Many times other variables besides the decision variables are introduced in the model to help formulate the objective function or the constraint equations. Such variables, which are not part of the original problem, are called auxiliary variables. We consider some problem formulation possibilities with binary variables and auxiliary variables next.

Either-Or Decision Variables

Problems that involve "yes-or-no decisions" or "on-or-off" variables can be modeled by allowing the variables to assume a binary value, 0 or 1. Thus the jth variable would be represented by X_j defined as

$$x_j = \begin{cases} 1, & \text{if decision } j \text{ is yes,} \\ 0, & \text{if decsion } j \text{ is no.} \end{cases}$$

Constraint Selection

Sometimes a model will include a set of m possible constraints for which any k of them must hold. The model then chooses the combination of k constraints that optimizes the objective function.

We denote the m possible constraints by the following functions:

$$f_1(x_1, x_2, ..., x_n) \leq b_1$$

$$f_2(x_1, x_2, ..., x_n) \leq b_2$$

$$\vdots$$

$$f_m(x_1, x_2, ..., x_n) \leq b_m$$

A set of m auxiliary binary variables $y_1, y_2, ..., y_m$ are introduced into the constraints to obtain this formulation:

$$f_1(x_1, x_2, ..., x_n) \leq b_1 + My_1$$

$$f_2(x_1, x_2, ..., x_n) \leq b_2 + My_2$$

$$\vdots$$

$$f_m(x_1, x_2, ..., x_n) \leq b_m + My_m$$

$$\sum_{i-1}^{m} y_i = m - k,$$

and y_i is a binary variable, for $i = 1, 2, ..., m$,

where the auxiliary variable yt is defined as:

$$y_i = \begin{cases} 1, & \text{if constraint } i \text{ holds,} \\ 0, & \text{if decision } i \text{ does not hold.} \end{cases}$$

and M is an extremely large (within the context of the problem) positive number. If y_i, $= 0$, then $My_i = 0$, which reduces the augmented constraint to its original form. But, when $y_i = 1$, then My_t makes the right-hand-side of the constraint inequality extremely large (unbounded within the context of the problem). Therefore, the corresponding constraint is effectively eliminated from the model.

Either-Or Constraints

A specific application to the constraint selection technique described above involves two constraints from which only one must hold. This requirement can be stated as
either

$$f_1(x_1, x_2, ..., x_n) \leq b_1$$

or

$$f_2(x_1, x_2, ..., x_n) \leq b_2.$$

The model formulation that achieves the desired effect is

$$f_1(x_1, x_2, ..., x_n) \leq b_1 + My_1$$

$$f_2(x_1, x_2, ..., x_n) \leq b_2 + My_2$$

and y is a binary varible.

CHAPTER 3

MODELS

3.1 Description of the model

We propse a model to use binary variables in a mixed-integer model to find optimal strategies. The object is to maximize the number of infected patients cured from the clinic perspective under a fixed program budget.

Based on CT and GC infection and coinfection rates, we analyzed the following five scenarios that differed by the pathogens tested for, and/or whether or not presumptive treatment is done: (1) screen for CT and treat those who test positive; (2) screen for GC and treat those who test positive; (3) screen for both CT and GC and treat those who test positive; (4) Screen for CT and treat those with positive CT results for both CT and GC;(5) Screen for GC and treat those with positive GC results for both GC and CT. For each of the five screening and treatment combinations, the model analyzed screening coverage for twelve population-groups divided by 3 age groups: (1) less than 20 years; (2) 20-24 years; and (3) more than 24 years; 4 race groups: (1) White; (2) Black; (3) Hispanic; (4)Other. six tests, and two treatments for each infection. The tests differed in performance on sensitivity and specificity or whether they were capable of detecting a single pathogen or both.

Currently, CDC mainly uses two pathogen-specific tests each for CT and GC and two combination tests for both CT and GC at the same time. For CT-specific tests, the commonly used nucleic acid hybridization test (Pace 2; Gen-Probe, San Diego, CA) and a nucleic acid amplification test (strand displacement amplification [SDA]; BDProbeTec, BD Biosciences, Sparks, MD) were modeled. For GC-specific tests, we modeled culture and a

nucleic acid amplification test (polymerase chain reaction [PCR]; Roche Diagnostic Systems, Branchburg, NJ). The two combination tests we modeled were the nucleic acid hybridization test, PACE 2C manufactured by Gen-Probe, and a bundled nucleic acid amplification test BDProbeTec CT/GC test by Becton Dickinson. We modeled two commonly-used CDC-recommended treatments for CT infection, (azithromycin and doxycycline). For GC infection, we modeled ceftriaxone and cefpodoxime. The use of cefpdoxime to treat for GC is not yet recommended by CDC. But with reported increasing resistance to fluoroquinolones and the discontinued production of cefixime (the only CDC-recommended oral treatment besides fluoroquinolones). some health departments have suggested the use cefpodoxime as an alternative oral treatment [17].

Without lose of generality, we address some extensions to the tests and treatments. We assumed that there were r screenings for CT, s screenings for GC, t combination tests for both CT and GC; and there are u treatments for CT, v treatments for GC. Those extentions would make this model more flexible and practical.

A number of simplifying assumptions were made in the model. First, we assumed that all women who visited the family planning clinic and were infected with CT and/or GC had no symptoms of infection. This simplify assumption was based on data from Philadelphia's publicty funded family planning program showing that <10% of women have symptoms of CT infection. Second, we assume all women who tested positive for CT infection and/or GC would be treated. Third, we considered strategies that could allow the screening and treatment of some or all population groups. Fourth, we assumed that either all or none of the women in each age-group will be screened. Fifth, we assumed that the return rate for treatment for women who tested positive for CT and/or GC is uniform for all age groups. Sixth, we assumed that the same test and treatment were offered to each visiting population group. Seventh, we assumed that no patient received more than one test or treatment for

CT and/or GC.

3.2 Data used in the model

We used data from published literature and from expert opinion from public health researchers where data was unavailable or insufficient(table 1). Group-specific infection rates of CT and GC were based on data from Philadelphia family planning clinics from 1996-1997, and from the CDC 2002 Sexually Transmitted Disease Surveillance Report [18]. Group-specific coinfection rates were based on a recent CDC study that examined GC infection rates and coinfection rates with CT [19]. Test costs included the cost of the test kits, reagents and labor time for collection and processing. We did not include the cost for a pelvic examination because such an examination is often conducted for reasons other than CT or GC screening. The cost of a pelvic examination was thus not considered as a direct screening-specific cost.

All costs published in previous literature were adjusted to 2003 US dollars using the medical component of the Consumer Price Index.

Table 3.1: Variables used in the optimal resource allocation model

Variable	CT baseline(range)	GC baseline(range)	Reference
Prevalence of infection			
white			
Among women aged <20 y	0.013-0.025	0.006-0.008	
Among women aged 20-24 y	0.013-0.025	0.006-0.008	
Among women aged >24 y	0.013-0.025	0.006-0.008	
black			
Among women aged <20 y	0.1-0.125	0.012-0.081	
Among women aged 20-24 y	0.1-0.125	0.012-0.081	
Among women aged >24 y	0.1-0.125	0.012-0.081	
hispanic			
Among women aged <20 y	0.023-0.030	0.004-0.020	
Among women aged 20-24 y	0.023-0.030	0.004-0.020	
Among women aged >24 y	0.023-0.030	0.004-0.030	
others			
Among women aged <20 y	0.023-0.030	0.004-0.020	
Among women aged 20-24 y	0.023-0.030	0.004-0.020	
Among women aged >24 y	0.023-0.030	0.004-0.030	
Sensitivity of tests			
Pace 2 CT	0.780	N/A^*	
BDProbeTecCT	0.928	N/A^*	

	Culture	N/A^*	0.850
	PCR	N/A^*	0.900
	Pace 2C Combo	0.78	0.810
	BDProbeTec CT/GC	0.928	0.966
Specificity of CT tests			
	Pace 2 CT	0.993	N/A^*
	BDProbeTecCT	0.981	N/A^*
	Culture	N/A^*	0.995
	PCR	N/A^*	0.990
	Pace 2C Combo	0.993	0.993
	BDProbeTec CT/GC	0.993	0.995
Cost of tests	Pace 2 CT	8.03	N/A^*
	BDProbeTecCT	9.42	N/A^*
	Culture	N/A^*	4.20
	PCR	N/A^*	9.26
	Pace 2C Combo	$4.31 + 8.03$	4.31+7.51
	BDProbeTec CT/GC	7.82	7.82
Effectiveness of treatment			
	Doxycycline	0.900	N/A^*
	Azithomycin	0.965	N/A^*
	Ceftriaxone	N/A^*	0.977
	Ciprofloxacin	N/A^*	0.972
	Cefpodoxime	N/A^*	0.965
Cost of treatment			
	Doxycycline	4.00	N/A^*

Azithomycin	9.50	N/A^*
Ceftriaxone	N/A^*	15.37
Ciprofloxacin	N/A^*	5.27
Cefpodoxime	N/A^*	7.32
Probabilites		
PID from untreated infection	0.21	0.21
Patient's return for treatment	0.81	0.81
Cost per case of PID	1434(1434-4131)	
Cost of treatment	14.00	

3.3 The mathematical model

To find an optimal strategy that maximizes the number of infected patients cured, we need to determinize which population group to screen, which testing technologies to use, and which treatments to use. We used a mixed-integer program to determin the optimal strategy. In the mixed-integer program, three sets of indexed binary variables were created: (1) x_i is a binary variable for population group, $x_i = 1$ if group i is selected and 0 otherwise. (2) y_j is a binary variable for screening, $y_j = 1$ if screening method j is selected and 0 otherwise; (3) x_{kl} for presumptive treatment. The first index i corresponds to population-group(i=1,2, ...,12); the index j corresponds to screen methods (i=1,2, ...,6); the index k corresponds to treatment for CT(k=1,2); The index l corresponds to treatment for GC(l=1,2), the notations and variables defined as below:

Notation:

- let $Sct1$ denotes the screening assay 1 for CT, $Sct2$ denotes the screening assay 2 for CT
- let $Sgc1$ denotes the screening assay 1 for GC, $Sgc2$ denotes the screening assay 2 for GC
- let $Scombo1$ denotes the screening assay 1 for both CT and GC, $Scombo2$ denotes the screening assay 2 for both CT and GC

The fifth index m corresponds to treatment, also there exists three conditions:

- let $Tct1$ denotes treatment 1 for CT, let $Tct2$ denotes treatment 2 for CT
- let $Tgc1$ denotes treatment 1 for GC, let $Tgc2$ denotes treatment 2 for GC

Variables:

1. There are 12 population groups and set $x_i = 1$ if group i is selected and 0 otherwise.

2. There are 6 screening methods: $Sct1$ (1), $Sct2$ (2), $Sgc1$ (3), $Sgc2$(4), $Scomb1$ (5), $Scomb2$ (6).

   Let $y_j = 1$ is screening j is selected and 0 otherwise.

3. There are 8 treatment methods: $Tct1$ (1,0), $Tct2$ (2,0), $Tgc1$ (0,1), $Tgc2$ (0,2), $Tctk + Tgcl = (k, l)$. for $k = 1, 2$ and $l = 1, 2$.

   Let $x_{kl} = 1$ if treatment (k, l) is selected and 0 otherwise.

Target Functions: the number of unit of diseases that would be cured

For $j = 1, 2$, means to use the screen assay 1 for CT or screen assay 2 for CT to screen all the population groups let

$$\begin{aligned} Cured_{ijkl} &= Prev(i,1) \cdot sen_j \cdot Eff(k) \\ &+ Prev(i,1) \cdot sen_j \cdot CoPrev(1,2) \cdot Sign(l) \cdot Eff(l) \\ &+ (1 - prev(i,1)) \cdot (1 - Spe_j) \cdot CoNonPrev(1,2) \cdot sign(l) \cdot Eff(l) \end{aligned}$$

Where $CoNonPrev(1,2) = \frac{prev(1,2) - Prev(i,1) \cdot CoPrev(1,2)}{1 - Prev(i,1)}$.

Because

$$\begin{aligned} Pop_i \cdot Prev(i,2) &= Pop_i \cdot Prev(i,1) \cdot Copre(1,2) \\ &+ Pop_i \cdot (1 - Prev(i,1)) \cdot CoNonPrev(1,2) \end{aligned}$$

then one can get

$$CoNonPrev(1,2) = \frac{prev(1,2) - Prev(i,1) \cdot CoPrev(1,2)}{1 - Prev(i,1)}$$

For $j = 3, 4$, means to use the screen assay 1 for GC or screen assay 2 for GC to screen the population groups we define $Cured_{ijkl}$ similarly. For $j = 3, 4$, let

$$\begin{aligned} Cured_{ijkl} &= Prev(i,2) \cdot sen_j \cdot Eff(l) \\ &+ Prev(i,1) \cdot sen_j \cdot CoPrev(2,1) \cdot Sign(k) \cdot Eff(k) \\ &+ (1 - prev(i,2)) \cdot (1 - Spe_j) \cdot CoNonPrev(2,1) \cdot sign(k) \cdot Eff(k) \end{aligned}$$

where

$$CoNonPrev(2,1) = \frac{prev(2,1) - Prev(i,2) \cdot CoPrev(2,1)}{1 - Prev(i,2)}$$

For $j = 5, 6$, means to use the combination screen assay1 or combination screen assay2 to screen both CT and GC, let

$$\begin{aligned} Cured_{ijkl} &= Prev(i,1) \cdot Sen(j,1) \cdot Eff(k) + \\ &+ Prev(i,2) \cdot Sen(j,2) \cdot Eff(l) \end{aligned}$$

Let

$$TCured = \sum_{ijkl} Pop_i \cdot ReProb(i,j) \cdot Cured_{ijkl} \cdot x_i \cdot y_j \cdot z_{k,l}$$

Cost Functions:

For $j = 1, 2$, let

$$
\begin{aligned}
Cost_{ijkl} &= Pop_i(BaseCost_j + VisitingCost) \\
&+ Pop_i(Prev(i,1) \cdot Sen_j + (1 - Prev(i,1))(1 - Spe_j)) \\
&\cdot (DrugCost(k))ReProb(i,1) + \\
&+ Pop_i(Proev(i,1)Sen_j + (1 - Prev(i,1))(1 - Spe_j)) \cdot \\
& sign(l) \cdot DrugCost(l) \cdot ReProb(i,1)
\end{aligned}
$$

For $j = 3, 4$, we define $Cost_{ijkl}$ similarly.

$$
\begin{aligned}
Cost_{ijkl} &= Pop_i(BaseCost_j + VisitingCost) \\
&+ Pop_i(Prev(i,2) \cdot Sen_j + (1 - Prev(i,2))(1 - Spe_j)) \\
&\cdot DrugCost(l)ReProb(i,2) + \\
&+ Pop_i(Proev(i,2)Sen_j + (1 - Prev(i,2))(1 - Spe_j)) \cdot \\
& sign(k) \cdot DrugCost(k) \cdot ReProb(i,2)
\end{aligned}
$$

For $j = 6$, let

$$
\begin{aligned}
Cost_{ijkl} &= Pop_i(BaseCost + VisitingCost) \\
&+ Pop_i(Prev(i,1)Sen(j,1) + (1 - Prev(i,1))(1 - Spe(j,1))) \\
& \cdot DrugCost(k) \cdot ReProb(i,1) \\
&+ Pop_i(Prev(i,2)Sen(j,2) + (1 - Prev(i,2))(1 - Spe(j,2))) \\
& \cdot DrugCost(l) \cdot ReProb(i,2)
\end{aligned}
$$

For $j = 5$, this is a special case, because this screen has additional cost, which used to identify what infection(s) the patients have been infected. We need to introduce some new

notation here

- let $BaseSen(1,0)$ denote the base sensitivity for CT
- let $BaseSen(0,1)$ denote the base sensitivity for GC
- let $BaseSen(1,1)$ denote the base sensitivity for testing the people who has both CT and GC

For $j = 5$, the cost is:

$$\begin{aligned}
Cost_{ijkl} &= Pop_i \cdot \{Prev(i,1) \cdot (1 - Coprev(1,2)) \cdot BaseSen(1,0) \\
&+ Prev(i,1) \cdot Coprev(1.2) \cdot BaseSen(1,1) \\
&+ Prev(i,2) \cdot (1 - Coprev(2,1)) \cdot BaseSen(0,1) \\
&+ [(1 - Prev(i,1)) - (1 - prev(i,1)) \cdot CoNoprev(1,2)] \cdot (1 - BaseSpe)\} \\
&\cdot (AddCost(1) + AddCost(2)) \\
&+ Pop_i(Prev(i,1)Sen(j,1) + (1 - Prev(i,1))(1 - Spe(j,1))) \\
&\cdot DrugCost(k) \cdot ReProb(i,1) \\
&+ Pop_i(Prev(i,2)Sen(j,2) + (1 - Prev(i,2))(1 - Spe(j,2))) \\
&\cdot DrugCost(l) \cdot ReProb(i,2)
\end{aligned}$$

Therefore the problem is

$\max TCured = \sum_{i,j,k,l} Cured_{ijkl} x_i y_j z_{l,m}$.

subject to

$$\sum_{ijkl} Cost_{ijk2} x_i y_j z_{l,m} \leq Budget.$$

$$\sum_{j=1}^{6} y_j \leq 1,$$

$$\sum_{k,l} z_{kl} = 1.$$

As we can see, this model could be caterogrize into the $0-1$ **Knapsack problem**(KP problem) for fixed j, k and l. The definition of 0-1 KP problem is: given a set of n items and a knapsack , with

$$p_j = profit \text{ of item } j$$

,

$$w_j = weight \text{ of item } j$$

,

$$c = capacity \text{ of item } j$$

,

select a subset of the items so as to

$$\text{maximize z= } \textstyle\sum_{j=1}^{n} p_j x_j$$

$$\text{subject to: } \textstyle\sum_{j=1}^{n} x_j \leq c$$

$$x_j = 0 \text{ or } 1,\ j \in N = 1, \ldots, n;$$

where

$$x_j = \begin{cases} 1, & \text{if item } j \text{ is selected,} \\ 0, & \text{otherwise.} \end{cases}$$

In the fifties, Bellman's dynamic programming theory produced the first algorithm to exactly solve the 0-1 KP problem. In the sixties, the dynamic programming approach to the KP and other KP-type problems was deeply investigated by Gilmore and Gomory. In 1967, Kolesar experimented with the branck-and-bound algorithm for the problem. In the seventies, the branch-and-bound algorithm was further developed, proving to be the only method capable of solving problems with a high number of variables.

In order to solve our practical problem, we developed a two-step algorithm, combining exhaustive and dynamic programming algorithms.

3.4 The Algorithm

In the seven assumptions provided previously, there is a very strong one that simplifies our model a lot. We assume that all population groups should take the same test and same treatment(s). It means for all population groups which are selected to attend the test and treatment(s), the test and treatment(s) should be all the same to every group.

For Screening and treatments, there are r test assays for CT, s for GC, t assays for combination tests. For treatments, there are u assays for treatments of CT, v assays for treatments for GC. Hence, there are uv combinations to choose one from the treatments of CT and one from the treatments of GC. The total combinations of CT screening and treatments is: $r(u + uv)$. For the same reason, the combination of GC screening and treatments is : $s(v + uv)$. If the population groups use combination tests for both CT and GC, the combination numbers are tuv. so the total combinations is :

$$r(u + uv) + s(v + uv) + tuv$$
$$= ru + sv + uv(r + s + t)$$

Obviously, the complexity of the exhaustive algorithm in this case is in term of r,s,u,t.

Considering the population group, there are 12 groups in our model. If each group chooses to go for test or not to go, then, this is a knapsack problem. The number of the population group's choices is 2^{12}. If there are more population groups, the combinations would increase exponentially. And the Knapsack problem is a NP-Complete, there is no known polynomial-time alogrithm for this type of question. Therefore, we need to use dynamic programming or a branch and bound algorithm to deal with it with the hope of reducing the complexity. Here we decide to put dynamic programming into use because the number variables in our model is not that high.

Therefore, the model could be split up into two parts, accordingly, We bring up a two-period algorithm to process it.

1). At the first period, we exhaust all the combinations of test and the treatments. The total combinations is $ru + sv + uv(r + s + t)$. The polynomial-time problem makes the exhaustive algorithm reasonable.

2). At the second period, we use dynamic programming algorithm to find out which group(s) should be enforced test and treatment(s). The Knapsack problem is a NP-complete problem. The complexity is non-polynomial.

The knapsack problem can be solved in pseudo-polynomial time using dynamic programming. The following depicts a dynamic programming solution for the unbounded knapsack problem.

Let the costs be $c_1, \ldots c_n$ and the corresponding values $v_1, \ldots v_n$. We wish to maximize total value subject to the constraint that total cost is less than C. Then for each $i \leq C$, define $A(i)$ to be the maximum value that can be attained with total cost less than or equal to i, clearly, $A(C)$ is the solution of the problem.

Define the $A(i)$ recursively as follows:

- $A(0) = 0$
- $A(i) = max\{v_j + A(i - c_j) \text{ for all } c_j \leq i\}$

Here the maximum of the empty set is taken to be zero. Tabulating the results from $A(0)$ up through $A(C)$ gives the solution. Since the calculation of each $A(i)$ involves examining n items (all of which have been previously computed), and there are C values of $A(i)$ to calculate, the running time of the dynamic programming solution is thus $O(nC)$.

In this paper, we performed a two period model of resources allocation model on a web-based application. The front-end webpage is written by Java Sever Page(JSP), which in charge of inputing data and output result. The back-end algorithm is written in Java. The user will get the data from the webpage and feedback an optimization allocation method.

This does not contradict the fact that the knapsack problem is NP-complete, since C, unlike n, is not polynomial in the length of the input to the problem. The length of the input to the problem is proportional to the number of bits in C, not to C itself.

A similar dynamic programming solution for the 0-1 knapsack problem also runs in pseudo-polynomial time[20]. As above, let the costs be $c_1, \ldots, c_n$ and the corresponding values $v_1, \ldots, v_n$. We wish to maximize total value subject to the constraint that total cost is less than C. Define a recursive function, $A(i, j)$ to be the maximum value that can be

attained with cost less than or equal to j using items up to i.

- $A(0, j) = 0$
- $A(i, 0) = 0$
- $A(i, j) = A(i - 1, j)$if $c_i > j$
- $A(i, j) = max(A(i - 1, j), v_i + A(i - 1, j - c_i)$ if $c_i \leq j)$

The solution can then be found by calculating $A(n, C)$. To do this efficiently we can use a table to store previous computations. This solution will therefore run in $O(nC)$ time and $O(nC)$ space, though with some slight modifications we can reduce the space complexity to $O(C)$.

CHAPTER 4

RESULTS AND DISCUSSIONS

4.1 Results

Table 4.1: Optimal Strategy

total budget	budget per patient	population group	test	treatment	total cured person
19,000	19.0	2	CULTURE	Doxy+Cip	72
19,100	19.1	2	CULTURE	Azi+Cip	73
19,680	19.68	2	CULTURE	Azi+Cef	74
20,140	20.14	2	CULTURE	Azi+Cef	74
20,150	21.25	2	APTIMACT/GC	Doxy+Cip	105
20,180	20.18	6	APTIMACT/GC	Azi+Cef	155
40,050	20.02	2,9	APTIMACT/GC	Doxy+Cip	164
40,500	20.25	2,6	APTIMACT/GC	Doxy+Cip	254
43,500	21.75	2,6	APTIMACT/GC	Azi+Cef	265
205,000	20.05	1,2,3,4 5,6,7,8 10,11	APTIMACT/GC	Azi+Cef	539
300,000	25,00	1-12	APTIMACT/GC	Azi+Cef	579

we applied our model to a hypothetical cohort of 1000 asymptomatic women evenly distributed among the population groups considered. The strategies that maximized the num-

ber of cures at different total budgets and per-patient budgets are presented in Table 2. In the Table 2 above, the index number 1-12 of the population groups stands for the group with different age and races, as the Table3 shows:

From the the Table 2 we can know, for example, under the total budget of $19,000, only population group 2 would choose to take part in the test and treatment. Then the per-patient budget is $19.00. Use Culture for the test assay, Doxy and Cip for the treatments. The number of cured people would be 72. Population group 2 stands for the group of black people with the age less than 20 years old. The infection rate of CT and GC ranges from 0.1-0.125 and 0.012 - 0.081 respectively, which are much higher than other groups. Therefore, when the budget is not enough, this group should get priority to be tested and treated. In addition, this group's infection rate of CT is much higher than the rate of GC, the optimal strategy is to choose test assays for CT rather than GC or the combined test because of the limited budget.

All of the optimal strategies listed in Table 2 could be classifed into two groups. When the total budget is limited, the best strategy is to choose the less expensive Culture as the test assay. When the budget is enough, we should use the expensive combined test-APTIMA CT/GC as the test assay.

Frome Table 2, one could see the overall trend is that if the total budget increases the total number of cured persons also increases. But when you considering the per patient budget, the situation becomes a little bit more complicated. With the increasing of the toal budget, the per patient budget goes up and downs. For example, when the total budget ranges from $125,000 to $165,000, the per patient budget drops from $20.8 to $18.30, but the total cured persons increases tremendously from 427 to 488.

From the Table 2, one could see the treatments would be Azi and Cef in most of the

occasions. In fact, treatments would not play as an important role in the budget planning. We can imagine that only a very small part of the whole population (sometimes the part is less than 1%) is the positive patients which need to be treated. Therefore the population to go for treatment would be very small. As a result, the money spent on treatment drugs would be very small. The key factor that would influence the budget planning tremendously is the test assay. All the population groups would go for screen, and everybody would be screened with a certain assay. Most of the budget would be invested in the test. Hence the decision of which test assay should be used is the critical factor.

From all the strategies listed in Table 2, we could not draw an conclusion about which one is better, to use a more sensitive and expensive test to screen fewer patients, or rather to use a relatively cheaper and less sensitive test to screen a greater number of patients. It depends on the data of the budget, infection rate and all kinds of characteristcs of the test assays and treatments. Based on our current data, we could say that to use less sensitive test to screen a great number of patients should be better when the total budget is limited.

This mathematical model was formulated by a web-based application. The front-end web page is written by Jave serve page. The back-end program is written by Java language. They are connected by Javabean. The whole framework of this web-application is based on **Model-Controller-View**(MVC) pattern, using MySQL as the database to store the data. We deploy the whole application on the sun-blade-100 workstation. The website's address:

http://131.96.241.29:8080/wx/project2.html. And the source code is attached in the appendix.

Table 4.2: population group

	white	black	hispanic	others

<20 years old	1	2	3	4
$20-24$ years old	5	6	7	8
>24 years old	9	10	11	12

4.2 Future Work

Although the models discussed in this thesis provide useful tools for clinicians in allocating resources for controlling C. *trachomatis*(CT) and Neisseria gonorrhoeae(GC) infections, they have several limitations and hence there still exist several opportunities to advance this study. Some of these limitations and opportunities for further work are discussed below.

Possible improvements

1. In our current web application, we make the categorization of the population group configurable. One can randomly divide the population into different groups by age and race. This makes the webpage very flexible and could meet the practical requirments of the user. With the exception for the population group, other parameters such like infections, screen assays, and treatments are all static. For example, one can not add or delete an infections or several infections one the webpage. The assays for screening and drugs for treatments are all constant. One could not add or delete them. Hence this web-based application is not expandable and could not satisfy user's practical requirments. The next step should be to make all the parameters such like infections, screen assays, and treatments drugs dynamic and configurable. Considering many branches of CDC would use this application for decision making later, each user would have its own profile for population groups, infections, screens and treatments. Besides making all of those parameters configurable, we should go further to adopt a database to store all user's profile data, which would be great convience for all the users.

2. We use the dynamic programming as the background algorithm. Compared with exhaustive mothod, dynamic programming saves time by the sacrifice of space. When the size of the problem becomes bigger, the memeory would be consumed very quickly or even the program could break down soon. As an improvement, the Branch and Bound algorithm is a good choice. For a given problem space an efficient division will divide the solution space into a small set containing high value (or low cost) solutions to be examined more closely and a larger set of their opposites (those to be ignored). If a partial solution cannot improve on the best, it is abandoned. Therefore, the computer is not necessary to store all data of combinations of population groups, screens and treatments. This saves space and time. Furthermore, This method naturally lends itself for parallel and distributed implementations. We could even implement the parallel computing and distributed computing. By parallel computing, a computing process would be divided into several threads,each thread would compute a subregions of the problem, hence a lot of time could be saved. By distributed computing, we could deploy the application onto several servers and each the server would collabrate for computation. The computing speed of the program would increase tremendously.

BIBLIOGRAPHY

[1] Centers for Disease Control and Prevention. Sexually Transmitted Disease Surveillance, 2001. Atlanta, GA: U.S. Department of Health and Human Services, Centers for Disease Control and Prevention, September 2002.

[2] Centers for Disease Control and Prevention.Sexally Transmitted Diseases Treatment Guidelines 2002. MMWR 2002; 51(NO.RR-6):32-36.

[3] Rein D, Kassler W, Irwin K, Rablee L. Direct Medical Cost of Pelvic Inflammtory Disease and its Sequelae: Decreasing, but Still Substantial. Obstet Gynecol 2000; 95:397-402

[4] American Social Health Association. Update on Fiscal Year 1994: Funding for the STD Program of the CDC and NIH. Sex Transm Dis 1994; 21:55-58

[5] Groseclose SL, Zaidi AA, DeLisle SJ, Levine WC, St Louis ME. Estimated Incidence and Prevalence of Genital Chlamydia *trachomatis*Infections in the United States, 1996. Sex Transm Dis 1999;26(6):339-44

[6] Mertz KJ, Voigt RA, Hutchins K, Levien WC, Jail STD Prevalence Monitoring Group. Findings from STD screening of adolescents and adults entering corrections facilities. Sex Transm Dis 2002; 29:834-39

[7] Farley TA, Cohen DA, Elkins W. A symptomatic sexually transmitted diseases: the case for mass screening. Am J Prev med(In Press)

[8] Centers for Disease Control and Prevention. Screening Tests to Detect Chlamydia trachomatis and Neisseria gonorrhoeae Infections-2002. MMWR 2002;52(No.RR-15):9-10,37.

[9] Commonwealth of Massachusetts Department of Public Health. Prevention and Management of Chlamydial Infections in Adolescents A Toolkit for CLinicians

[10] California Department of Health Services Sexually Transmitted Diseases (STD) Control Branch and the California STD Controllers Association. Guidelines for the Treatment of Chlamydia and Gonorrhea Cases and Exposed Sexual Partners by Health Department Staff in Non-Clinical Settings.

[11] Jones, Catherine A. PhD ; Knaup, Richard C. BA; Hayes, Mary BS; Stoner, Bradley P. MD, PhD Urine Screening for Gonococcal and Chlamydial Infections at Community-Based Organizations in a High-Morbidity Area.

[12] Tao G, Gift T, Walsh C, Irwin K, Kassler W. Optimal Resource Allocation for Curing Chlamydia trachomatis Infection among Asymptomatic Women of Clinics Operating on a Fixed Budget. Sex Transm Dis 2002;29(11):703-709

[13] Goeller B, the PAWN team. Planning the Neitherland's water resources. Interfaces 1985;15(1):3-33

[14] Kaplan EH, O'Keefe E. Let the Needles Do the Talking! Evaluating the New haven Needle Exchange. Interface 1993;23(1):7-26

[15] Warren A, DeWitt CE, Brenner DA, Ladson LS, Melheim SA, OMEGA: an improved gasoline blending system for Texaco. Interfaces 1989;19(1):85-101

[16] Fourer R. (August 2001) Linear programming software survey. OR/MS Today. http://lionhrtpub.com/orms/orms-8-01/survey/LP/LP-survey.html.

[17] Centers for Disease Control and Prevention.Oral Alternatives to Cefixime for the Treatment of Uncomplicated Neisseria gonorrhoeae Urogenital Infections. http://www.cdc.gov/std/treatment/Cefixime.htm

[18] Centers for Disease Control and Prevention. STD Surveillance 2002. http://www.cdc.gov/std/stats02/TOC2002.htm

[19] RA McDonald, J Pfisher.Effects of the Transition to Chlamydia Nucleic Acid Amplification Testing in a Public Health Screening Program

[20] Silvano Martello; Paolo Toth (1990). Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons. ISBN 0-471-92420-2.

APPENDIX

Code: Knapasack Algorithm

```
import javax.swing.UIManager;

/**
 * @author Wei Xin.
 */
public class Knapsack2 {
  static int v=0;
  /**
   * Calculates the optimal vector.
   * @param w are the weights.
   * @param p are the prices.
   * @param n is the number of items.
   * @param c is the capacity.
   * @param m are the coefficients of the constraints.
   * @param x is the resulting vector.
   */
  public static void knapsack2(int[] w, int[] p, int n,
                               int c, int[][] m, boolean[] x)
  {
    int i,j,k;
```

```
for(i=0;i<=n;i++)
{
  m[i][0] = 0;
  x[i] = false;
}
for(i=0;i<=c;i++){
  m[0][i]=0;
}


for(i=1;i<=n;i++)
{
  for(j=1;j<=c;j++)
  {
    m[i][j]=m[i-1][j];
    if((j>=w[i-1])&&((m[i-1][j-w[i-1]]+p[i-1])>m[i-1][j]))
      m[i][j]=m[i-1][j-w[i-1]]+p[i-1];
  }
}
j=c;
for(i=n;i>0;i--)
{
  if(m[i][j]>m[i-1][j])
  {
    x[i-1]=true; j=j-w[i-1];
  }
```

```
  }
  v=m[n][c];
}
/**
 * Get the result.
 * @return the result.
 */
public int get_result()
{
  return v;
}
/**
 * Prints the result.
 */
public static void printinfo()
{
  System.out.println(v);
}
/**
 * This is the main procedure.
 * @param args are the arguments.
 */
public static void main(String[] args)
{
  int[] ww = {2,2,3,5,4};
```

```
    int[] pp = {6,3,5,4,6};
    boolean[] xx = new boolean[6];
    int[][] mm = new int[11][11];
    int i;
    knapsack2(ww,pp,5,10,mm, xx);
    printinfo();
    for(i=0;i<5;i++)
      System.out.println(xx[i]);

  }
}
```

Code: calculate the coeffcients

```
package user7;
import java.text.*;
import java.util.*;

public class Coef{
int pop;                          //population
double upre;                      //upper prevalence
double sen;                       //sensitivity
double spe;                       //specificity
double bcost;                     //basecost
double acost;                     //additional cost
```

```
double dcost;                       //drug cost
double vcost;                       //visit cost
double eff;                         //effectiveness
double rp;                          //return probability
double copre;                       //coprevelance
double budget;                      //budget

double scured;
double mcured;
double tcost;
double drcost;
double mcost;

public Coef(int popi, double uprei, double seni, double spei, double bcosti, double acos
                      double dcosti, double vcosti, double effi, double rpi, double copr
{
   pop   = popi;
upre  = uprei;
sen   = seni;
spe   = spei;
bcost = bcosti;
acost = acosti;
dcost = dcosti;
vcost = vcosti;
eff   = effi;
```

```
rp    = rpi;
copre = coprei;
}
public void set_Scured()
{
scured = pop*upre*sen*eff*rp;  //calculate the CTcured & GCcured
}
public void set_Mcured()
{
mcured = pop*upre*sen*copre*eff*rp;  //CTinGCcured and GCinCTcured
}
public void set_Tcost()
{
tcost = pop*(bcost+(upre*sen*acost));  //CTtestcost and GCtestcost
}
public void set_Drcost()
{
drcost = pop*(upre*sen+(1-upre)*(1-spe))*(dcost+vcost)*rp;  //CTdrug cost and GCdrugcost
}
public void set_Mcost()
{
mcost = pop*dcost*rp*(upre*sen*copre+(1-upre)*(1-spe));     //CTinGCcost and GCinCTcost
}
public double get_Scured()
{
```

```
return scured;
}
public double get_Mcured()
{
return mcured;
}
public double get_Tcost()
{
return tcost;
}
        public double get_Drcost()
{
return drcost;
}
public double get_Mcost()
{
return mcost;
}
 public void printInfo()
{
this.set_Scured();
this.set_Mcured();
this.set_Tcost();
this.set_Drcost();
this.set_Mcost();
```

```
DecimalFormat df=new DecimalFormat("$#,###.##");
                System.out.println("singlecured: \t" +df.format(scured));
                System.out.println("mixedcured: \t" +df.format(mcured));
                System.out.println("testcost: \t" +df.format(tcost));
                System.out.println("drugcost: \t" +df.format(drcost));
                System.out.println("mcost: \t" +df.format(mcost));
}
public static void main(String[] args)
{
Coef coef1 = new Coef(1000, 0.125, 0.78, 0.993, 4.31, 8.03, 9.5, 14, 0.965, 0.81, 0.45);
        coef1.printInfo();
}
}
```

Code: Main Program(just part of them)

```
<%@ page import="user7.Coef" %>

<%@ page import="user6.Coefpage2" %>

<%@ page import="user8.Knapsack2" %>



<html>
```

```
<title>the setup of self-defined class and object</title>
<style type="text/css">
<!--
.style3 {
font-size: 16px;
font-weight: bold;
color: #FF0000;
}
-->
</style>
<body>

<%

  <%
   Coefpage2 coef2 = new Coefpage2(vcost1, rp1, budget1);
  coef2.setPop();
  coef2.getPop();

  %>




<%
```

```
double CTcured[][][] = new double[12][2][2];
double GCinCTcured[][][] = new double[12][2][2];
double CTtestcost[][][] = new double[12][2][2];
double CTdrugcost[][][] = new double[12][2][2];
double GCinCTcost[][][] = new double[12][2][2];

double GCcured[][][] = new double[12][2][2];
double CTinGCcured[][][] = new double[12][2][2];
double GCtestcost[][][] = new double[12][2][2];
double GCdrugcost[][][] = new double[12][2][2];
double CTinGCcost[][][] = new double[12][2][2];

double ComboCTcured[][][] = new double[12][2][2];
double ComboGCinCTcured[][][] = new double[12][2][2];
double ComboCTtestcost[][][] = new double[12][2][2];
double ComboCTdrugcost[][][] = new double[12][2][2];
double ComboGCinCTcost[][][] = new double[12][2][2];

double ComboGCcured[][][] = new double[12][2][2];
double ComboCTinGCcured[][][] = new double[12][2][2];
double ComboGCtestcost[][][] = new double[12][2][2];
double ComboGCdrugcost[][][] = new double[12][2][2];
double ComboCTinGCcost[][][] = new double[12][2][2];

int[][] TotalCTcured = new int[12][12];
```

```
 int[][] TotalCTcost = new int[12][12];
 int[][] TotalGCcured = new int[12][12];
 int[][] TotalGCcost = new int[12][12];
 int[][] TotalCombocured = new int[12][8];
 int[][] TotalCombocost = new int[12][8];

int[] Finalcured = new int[32];
 int[] Finalcost = new int[32];
 int j=0;

 int MaxCured = 0;

 int   gs1, gs2;
 int i, k, l, r,  n;
 n = r = 0;

 gs1 = gs2 = 0;

for (i = 0; i < 12; i++)
 {
   j = 0;
   for (k = 0; k < 2; k++)
     for (l = 0; l < 2; l++) {
       Coef coefct = new Coef(pop[i], cupre[i], ctsen[k], ctspe[k],
                              ctbcost[k], ctacost[k],
```

```
                        ctdrcost[l], vcost1, cteff[l], rp1, copre[0]);
coefct.set_Scured();
coefct.set_Mcured();
coefct.set_Tcost();
coefct.set_Drcost();
coefct.set_Mcost();

CTcured[i][k][l] = coefct.get_Scured();
GCinCTcured[i][k][l] = coefct.get_Mcured();
CTtestcost[i][k][l] = coefct.get_Tcost();
CTdrugcost[i][k][l] = coefct.get_Drcost();
GCinCTcost[i][k][l] = coefct.get_Mcost();

Coef coefgc = new Coef(pop[i], gupre[i], gcsen[k], gcspe[k],
                        gcbcost[k], gcacost[k],
                        gcdrcost[l], vcost1, gceff[l], rp1, copre[1]);
  coefgc.set_Scured();
  coefgc.set_Mcured();
  coefgc.set_Tcost();
  coefgc.set_Drcost();
  coefgc.set_Mcost();

  GCcured[i][k][l] = coefgc.get_Scured();
  CTinGCcured[i][k][l] = coefgc.get_Mcured();
  GCtestcost[i][k][l] = coefgc.get_Tcost();
```

```
        GCdrugcost[i][k][l] = coefgc.get_Drcost();
        CTinGCcost[i][k][l] = coefgc.get_Mcost();



      TotalCTcured[i][j] = (int) CTcured[i][k][l] +
          (int) GCinCTcured[i][k][l];
      TotalCTcost[i][j] = (int) CTtestcost[i][k][l] +
          (int) CTdrugcost[i][k][l] + (int) GCinCTcost[i][k][l];
      TotalGCcured[i][j] = (int) GCcured[i][k][l] +
          (int) CTinGCcured[i][k][l];
      TotalGCcost[i][j] = (int)GCtestcost[i][k][l] +
          (int) GCdrugcost[i][k][l] + (int) CTinGCcost[i][k][l];
      j++;
    }
 }

for (i = 0; i < 12; i++)
 { j=4;
   for (k = 0; k < 2; k++)
     for (l = 0; l < 2; l++)
       for (r = 0; r < 2; r++) {
         TotalCTcured[i][j] = (int) CTcured[i][k][l] +
             (int) GCinCTcured[i][k][l]+(int) GCcured[i][k][r] +
           (int) CTinGCcured[i][k][r];
         TotalCTcost[i][j] = (int)CTtestcost[i][k][l]+
```

```
                                    (int) CTdrugcost[i][k][l]
            +(int) GCinCTcost[i][k][l]+(int) GCdrugcost[i][k][r]
            +(int) CTinGCcost[i][k][r];
        TotalGCcured[i][j] = (int) GCcured[i][k][l] +
            (int) CTinGCcured[i][k][l]+(int) CTcured[i][k][r] +
          (int) GCinCTcured[i][k][r];
      TotalGCcost[i][j] =TotalCTcost[i][j] = (int)GCtestcost[i][k][l]+
                                (int)GCdrugcost[i][k][l]
          +(int) CTinGCcost[i][k][l]+(int)CTdrugcost[i][k][r]
          +(int)GCinCTcost[i][k][r];


        j++;


      }
 }


for (i = 0; i < 12; i++) //calculate the combo
 { j=0;
   for (k = 0; k < 2; k++)
     for (l = 0; l < 2; l++)
       for (r = 0; r < 2; r++) {
         Coef coefcomboct = new Coef(pop[i], cupre[i], combo_ctsen[k],
                                     combo_ctspe[k], combo_ctbcost[k],
                                     combo_ctacost[k],
                                     ctdrcost[l], vcost1, cteff[l], rp1,
```

```
                            copre[0]);
coefcomboct.set_Scured();
coefcomboct.set_Mcured();
coefcomboct.set_Tcost(); //the CT coeff in Combo
coefcomboct.set_Drcost();
coefcomboct.set_Mcost();

ComboCTcured[i][k][l] = coefcomboct.get_Scured();
ComboGCinCTcured[i][k][l] = coefcomboct.get_Mcured();
ComboCTtestcost[i][k][l] = coefcomboct.get_Tcost();
ComboCTdrugcost[i][k][l] = coefcomboct.get_Drcost();
ComboGCinCTcost[i][k][l] = coefcomboct.get_Mcost();

Coef coefcombogc = new Coef(pop[i], gupre[i], combo_gcsen[k],
                            combo_gcspe[k], combo_gcbcost[k],
                            combo_gcacost[k],
                            gcdrcost[l], vcost1, gceff[l], rp1,
                            copre[1]);
coefcombogc.set_Scured();
coefcombogc.set_Mcured();
coefcombogc.set_Tcost(); //the GC coeff in Combo
coefcombogc.set_Drcost();
coefcombogc.set_Mcost();

ComboGCcured[i][k][l] = coefcombogc.get_Scured();
```

```
            ComboCTinGCcured[i][k][l] = coefcombogc.get_Mcured();
            ComboGCtestcost[i][k][l] = coefcombogc.get_Tcost();
            ComboGCdrugcost[i][k][l] = coefcombogc.get_Drcost();
            ComboCTinGCcost[i][k][l] = coefcombogc.get_Mcost();

            TotalCombocured[i][j] = (int)ComboCTcured[i][k][l]
                +(int)ComboGCinCTcured[i][k][l]
                +(int)ComboGCcured[i][k][r]
                +(int)ComboCTinGCcured[i][k][r];
            TotalCombocost[i][j]= (int)ComboCTtestcost[i][k][l]
                + (int)ComboCTdrugcost[i][k][l] + (int)ComboGCinCTcost[i][k][l]
                + (int)ComboGCtestcost[i][k][l]+ (int)ComboGCdrugcost[i][k][r]
                + (int)ComboCTinGCcost[i][k][r];

                j++;

          }
     }

  %>

<%
```

```
     for(j=0;j<12;j++)
{
  n = 0;
  {
    for (i = 0; i < 12; i++) {
      Finalcured[n] = TotalCTcured[i][j];
      Finalcost[n] = TotalCTcost[i][j];
      n++;
    }

    int[][] mm = new int[13][ (int)budget1 + 1];
    boolean[] x = new boolean[13];
    Knapsack2 kp = new Knapsack2();
    kp.knapsack2(Finalcost, Finalcured, 12, (int)budget1, mm, x);
    gs1=kp.get_result();
    //kp.printinfo();

    if (MaxCured < gs1) {
      MaxCured = gs1;
      r=j;
      gs2=1;
    }

  }
```

```
}

for(j=0;j<12;j++)
{
  n = 0;
  {
    for (i = 0; i < 12; i++) {
      Finalcured[n] = TotalGCcured[i][j];
      Finalcost[n] = TotalGCcost[i][j];
      n++;
    }

    int[][] mm = new int[13][ (int)budget1 + 1];
    boolean[] x = new boolean[13];
    Knapsack2 kp = new Knapsack2();
    kp.knapsack2(Finalcost, Finalcured, 12, (int)budget1, mm, x);
    gs1=kp.get_result();
    //kp.printinfo();

    if (MaxCured < gs1) {
      MaxCured = gs1;
      r=j;
      gs2=2;
    }
```

```
  }
}

for(j=0;j<8;j++)
{
  n = 0;
  {
    for (i = 0; i < 12; i++) {
      Finalcured[n] = TotalCombocured[i][j];
      Finalcost[n] = TotalCombocost[i][j];
      n++;
    }

    int[][] mm = new int[13][ (int)budget1 + 1];
    boolean[] x = new boolean[13];
    Knapsack2 kp = new Knapsack2();
    kp.knapsack2(Finalcost, Finalcured, 12, (int)budget1, mm, x);
    gs1=kp.get_result();
    //kp.printinfo();

    if (MaxCured < gs1) {
      MaxCured = gs1;
      r=j;
      gs2=3;
    }
```

```
    }
  }



%>



</body>
</html>
```