T.R.
TURKISH NAVAL ACADEMY
NAVAL SCIENCE AND ENGINEERING INSTITUTE
DEPARTMENT OF INDUSTRIAL ENGINEERING

MAXIMAL COVERING LOCATION OF INTERCEPTOR BAYWATCH VESSELS

A MASTER THESIS

ÜMİT TANER TUZLACI

Advisor: Assoc.Prof. Hakan Tozan

İstanbul, 2013

T.R.
TURKISH NAVAL ACADEMY
NAVAL SCIENCE AND ENGINEERING INSTITUTE
DEPARTMENT OF INDUSTRIAL ENGINEERING

MAXIMAL COVERING LOCATION OF INTERCEPTOR BAYWATCH VESSELS

A MASTER THESIS

ÜMİT TANER TUZLACI

Advisor: Assoc.Prof. Hakan Tozan

İstanbul, 2013

MAXIMAL COVERING LOCATION OF
INTERCEPTOR BAYWATCH VESSELS

ÜMİT TANER TUZLACI

Submitted in partial fulfillment of the requirement for degree of

MASTER OF SCIENCE IN NAVAL OPERATIONS RESEARCH

Author: ----------
Ümit Taner Tuzlacı

Defense Date: 3rd June 2013

Approved by:

Assoc.Prof. Hakan Tozan (Thesis Advisor)

Prof. Emin Arca (Defense Committee Member)

Prof. Arif Nihat Güllüoğlu (Defense Committee Member)

DEDICATION

To my lovely wife Melek

and

my unique son Uğur.

ACKNOWLEDGEMENT

My advisor, Assoc.Prof. Hakan Tozan, deserves the best regards for his great guidance and support. Without his instructions this thesis would not be accomplished.

Besides, I would like to render my special thanks to,

My dear wife firstly, for her tolerance and support during my education,

My lecturers, Tevfik Altınalev, Asst. Prof. Murat Günal and Asst. Prof. İlker Akgün for their valuable contributions to my academic development,

My Institute Director, Dr. Mustafa Karadeniz for his moral support,

My colleagues Emre İrfanoğlu, Fatih Çelik, and Evren Olcaytu for their sincere fellowship.

DISCLAIMER STATEMENT

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Turkish Naval Forces, Turkish Naval Academy and Naval Science and Engineering Institute.

ÖNLEYİCİ DENİZ DEVRİYE VASITALARININ EN FAZLA KAPLAYARAK KONUŞLANDIRILMASI

Ümit Taner Tuzlacı

Deniz Harekât Analizi Yüksek Lisans Tezi, 2013

Danışman: Doç.Dr. Hakan Tozan

Anahtar Kelimeler: Küme kaplama problemi, maksimum kaplama yer seçimi problemi, tamsayılı programlama, genetik algoritma, önleyici deniz devriye vasıtası.

ÖZETÇE

Küme kaplama problemleri son yıllarda üzerinde yoğun çalışılan konulardan birisi haline gelmiştir. Bazı problemlerin çözümünde küme kaplama modelleri kullanılmış ve uygun çözüm yöntemleri geliştirilmiştir.

Bu çalışmada ele alınan problem "önleyici deniz devriye vasıtalarının en fazla kaplayarak konuşlandırılması" dır. Bu problem, küme kaplama probleminin özel bir hali olan maksimum kaplama yer seçimi problemi olarak ele alınabileceği değerlendirilmiştir. Maksimum kaplama yer seçimi modeli baz alınarak problemin kendine özgü kısıtları da entegre edilerek uygun bir matematiksel model elde edilmiştir.

Geliştirilen modelde belli sayı ve özellikteki vasıtanın önceden belirlenen bölgelere öngörülen kısıtları sağlayarak, maliyeti minimize edecek şekilde atanması amaçlanmıştır. Geliştirilen model, MATLAB Global Optimization Toolbox'ı kullanılarak genetik algoritma ile de çözülmüş, optimal ve genetik algoritma çözümleri analiz edilmiştir.

MAXIMAL COVERING LOCATION OF

INTERCEPTOR BAYWATCH VESSELS

Ümit Taner Tuzlacı

Naval Operations Research Master of Science Thesis, 2013

Advisor: Assoc.Prof. Hakan Tozan

Key Words: Set covering problem, maximal covering location problem, integer programming, genetic algorithm, interceptor baywatch vessel.

ABSTRACT

Set covering problems has become one of widely studied subjects in recent years. Set covering models used in the solution of some problems and appropriate solutions have been developed.

In this thesis, the problem "maximal covering location of interceptor baywatch vessels" is dealt. This problem is evaluated as the maximal covering location problem which is a special case of set covering problems. An appropriate mathematical model based on maximal covering location model is obtained by integrating unique constraints of the problem.

In the developed model, certain number and qualification of ships are intended to appoint certain pre-defined ports providing the prescribed constraints in order to minimize the cost. The model is also solved with genetic algorithm by using MATLAB Global Optimization Toolbox, optimal and genetic algorithm solutions are compared with analysis.

Contents

List of Figures

List of Tables

1 Introduction

An overview on the thesis, the motivation for doing this study and the progress of the thesis are explained in the following chapter.

1.1 Overview

This study is actually based on maximal covering location problem (MCLP) which is a special case of set covering problem (SCP). The detailed literature about the covering problem and genetic algorithm is given in chapter 2.

To date, scarce application of the MCLP to locate ships for maximal covering has appeared in the literature. In the context of this study, it is shown that covering models can result in savings in terms of cost of operations and decreased response times.

Covering models is used for the location of ambulance bases in rural areas, the location of emergency warning sirens generally in the literature. A few studies about coverage of seas are encountered during literature review.

In this thesis, a new mathematical model is developed in order to locate interceptor baywatch vessels for maximal covering of any particular sea area. Optimal and genetic algorithm solutions are applied to problem. In this location model when a new illegal event or a help call appears in the sea area, an interceptor baywatch vessel will certainly interfere in an hour.

The aim of the model is to locate interceptor baywatch vessels for maximal covering of any particular sea area. Therefore interceptor baywatch vessels are assigned appropriate ports in order to cover all the sea area.

In Chapter 3, a mathematical model is developed. In the model, certain number and qualification of ships are intended to appoint certain pre-defined ports providing the prescribed constraints in order to minimize the cost. In order to develop the model, missions and features of interceptor baywatch vessels are considered. Therefore, the missions of interceptor baywatch vessels and the features of interceptor baywatch vessels are explained in chapter 3.

Solution methodology is given in chapter 4. How optimal and genetic algorithm solutions are done is explained in that chapter.

In Chapter 5, experimental study is explained. Real situations and features of the Turkish interceptor baywatch vessels and features of the ports are used for experimental study. Optimal and genetic algorithm solutions are compared at the end.

It is obvious that GA solution is much faster than optimal solution. GA solution is the same as optimal solution in some problem sizes. For others, GA solutions are reasonably close to optimal solutions.

1.2 Motivation

Since the beginning, the Turkish interceptor baywatch vessels are not located in the light of scientific data. Therefore, no knowledge of how much money and time are wasted by not locating the vessels scientifically.

The aim of this thesis is to develop a new model in order to locate interceptor baywatch vessels for maximal covering of any particular sea area. In this location model when a new illegal event or a help call appears in the sea area, an interceptor baywatch vessel will certainly interfere in an hour.

In fact, interceptor baywatch vessels are assigned appropriate ports in order to cover all the particular sea area. While assigning the ships, it is considered features of ships and ports.

1.3 Progress of the Thesis

A continuous study is done while preparing the thesis. Problems and solutions to them are performed step by step.

Outline of the thesis is shown in Figure 1-1 below.

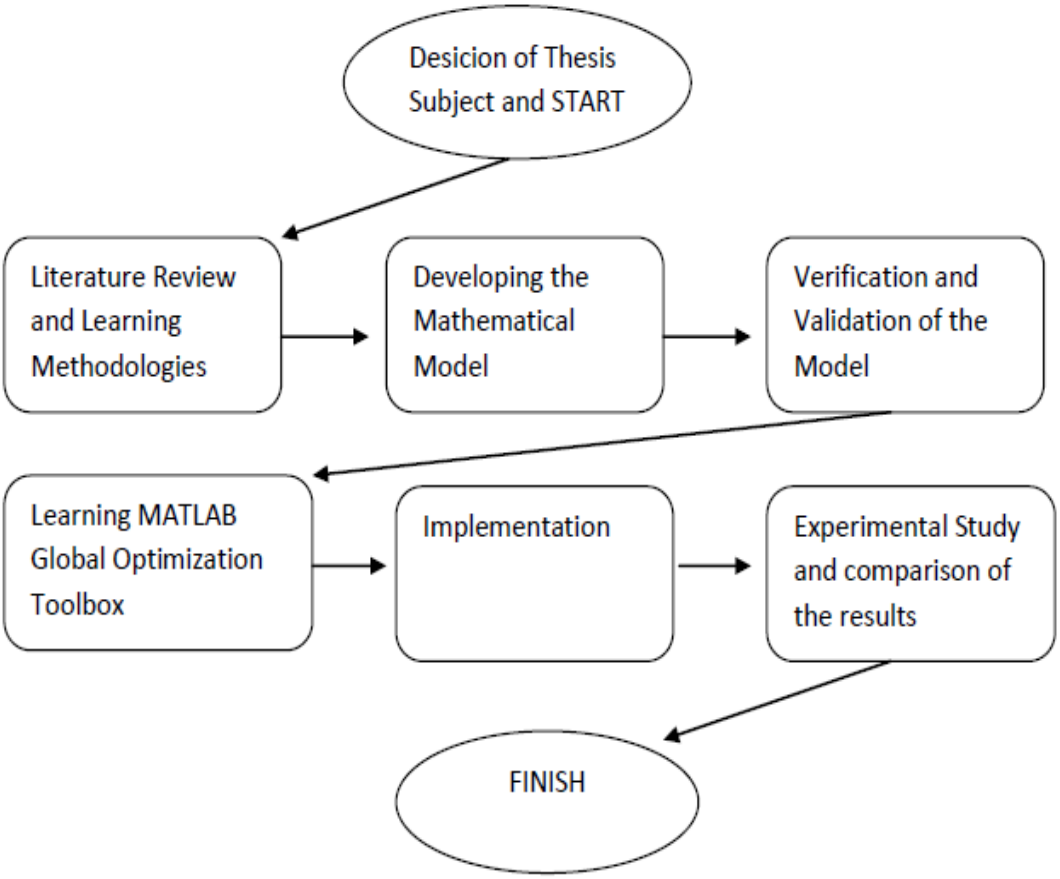

Figure 1-1: Progress of the Thesis

First, decision of the thesis subject is made and then studies are started. Then, literature review is done by surveying articles and thesis. Covering models, set covering problem, maximal covering location problem and genetic algorithm are researched. At the same time, solution methodologies are studied and learned.

Next, the problem is described as the maximal covering location problem. An appropriate mathematical model based on maximal covering location model is developed by integrating unique constraints of the problem.

Afterwards, verification and validation of the mathematical model is done. It is shown that correct assignments are taken from solvers for different sizes of problems.

Later on, MATLAB is selected as solver and it is started learning MATLAB Global Optimization Toolbox. Then, optimal and genetic algorithm solutions are developed in MATLAB Global Optimization Toolbox.

At the end, experimental study for real situations, features of the Turkish interceptor baywatch vessels and features of the ports is done. And, optimal and genetic algorithm solutions are compared.

2 Literature Review

Literature review of covering models, set covering problem (SCP), maximal covering location problem (MCLP) and genetic algorithm (GA) is expressed in the following chapter.

2.1 Covering Models

The main concept of covering problems is originally expressed by Toregas et al. (1971) and then it is extended by Berlin and Liebman (1974). In a classic covering problem, a demand point is called covered if it is within a defined distance, covering radius, from one of the closest facilities (Jabalamelia et al., 2011).

Covering models is used for location of emergency warning sirens, location of ambulance bases, ecological reserve selection and the location of retail facilities (Curtin et al., 2007). It is obvious that improving of covering models can result in savings in terms of cost of operations (Galvao et al., 1996).

While covering models are not new, they have always been very attractive for research. This is due to its applicability in real-world life, particularly for service and

emergency facilities. In some covering problems, a demand point should be served by at least one facility within a given critical distance (not necessarily the nearest facility). In most of the covering problems, customers take services by facilities depending on the distance between the customer and the facilities. The customer can take service from each facility which its distance from customer is equal or less than a defined number (Farahani et al., 2012).

Many of the problems like determining the number and locations of public schools, police stations, libraries, radar installations, hospitals, public buildings, military vases, post offices, branch banks and waste-disposal facilities can be modeled as covering problems.

Schilling et al. (1993) categorize models which use the notion of covering in two categories: (1) Set Covering Problem (SCP) where coverage is required and (2) Maximal Covering Location Problem (MCLP) where coverage is optimized. For each category, they ensure taxonomy according to topological structure, nature of need, characteristic of facility to be sited and application in public or private sectors. Also, based on solution methods of the models – either optimal or heuristic – a classification is proposed.

Basic studies on covering models are shown in Table 2-1 below.

Table 2-1: Basic Studies on Covering Models (Farahani et al., 2012)

Source title	Author name	Year	Document type	Subject area
European Journal of Operational Research (30)	ReVelle, C. (14)	2010 (165)	Article (1072)	Engineering (409)
Computers and Operations Research (28)	Berman, O. (12)			
Lecture Notes in Computer Science Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics (27)	Church, R. L. (11)			Earth and Planetary Sciences (326)
	Murray, A. T. (11)	2009 (178)	Conference Paper (348)	
Astrophysical Journal (23)	Marianov, V. (8)			Computer Science (253)
Proceedings of SPIE the International Society for Optical Engineering (18)	Drezner, Z. (7)			
	Krass, D. (7)	2008 (178)	Review (40)	Environmental Science (231)
Astronomy and Astrophysics (16)	Lowe, T. J. (7)			
Journal of Biogeography (16)	Tamir, A. (6)			Mathematics (172)
Journal of the Operational Research Society (15)	Galvao, R. D. (6)	2007 (128)	Article in Press (26)	
Global Ecology and Biogeography (14)	Serra, D. (6)			Social Sciences (160)
Annals of Operations Research (13)	Wesolowsky, G. O. (5)			
Forest Ecology and Management (11)	Laporte, G. (5)	2006 (103)	Conference Review (6)	Decision Sciences (159)
Journal of Geophysical Research D Atmospheres (9)	Kara, B. Y. (5)			
Socio Economic Planning Sciences (9)	Francis, R. L. (5)			Agricultural and Biological Sciences (159)
Atmospheric Environment (9)	Plastria, F. (5)	2005 (115)	Business Article (3)	
Location Science (8)	Lorena, L. A. N. (5)			Medicine (128)
Transportation Research Record (7)	Kraemer, S. B. (4)			
Papers in Regional Science (7)	Batta, R. (4)	2004 (93)	Short Survey (3)	Physics and Astronomy (111)
Computers and Industrial Engineering (7)	Hoefer, M. (4)			
Networks (7)	Crenshaw, D. M. (4)			Biochemistry, Genetics and Molecular Biology (101)
Proceedings of the National Academy of Sciences of the United States of America (7)	DeLand, M. T. (4)	2003 (66)	Note (1)	
	Drezner, Z. (4)			
Field Crops Research (7)	Gencer, C. (4)			Business, Management and Accounting (58)
Advances in Space Research (6)	Gerrard, R. A. (4)	2002 (62)	Report (1)	
International Journal of Climatology (5)	Emir-Farinas, H. (4)			
Pure and Applied Geophysics (5)	Van Dishoeck, E. F. (3)	(58)		Neuroscience (35)
Lecture Notes in Computer Science (5)	Vanhaverbeke, L. (3)	1999 (51)		
Annales Geophysicae (5)	Morabito, R. (3)			Economics, Econometrics and Finance (23)
IIE Transactions Institute of Industrial Engineers (5)	Brandenberg, R. (3)			
International Journal of Heat and Mass Transfer (4)	Schobel, A. (3)	1998 (55)		Chemistry (19)
Operations Research Letters (4)	Bottino, A. (3)			
Transportation Science (4)	Blake, G. A. (3)			
Icarus (4)	Schierbeck, J. (3)	1997 (52)		Immunology and Microbiology (18)
Journal of Vegetation Science (4)	Williams, J. C. (3)			
Theoretical and Applied Genetics (4)	Kotha, S. (3)			Multidisciplinary (16)
Annals of Glaciology (4)	Wu, N. W. (3)	1996 (49)		Health Professions (15)
Journal of Hydrology (4)	Gibson, A. (3)			
Water Science and Technology (4)	Yang, C. (3)			
Journal of Molecular Biology (4)	Mahlooji, H. (3)	1995 (22)		Psychology (13)
Monthly Weather Review (4)	Laurentini, A. (3)			
Monthly Notices of the Royal Astronomical Society (4)	Roth, L. (3)			Pharmacology, Toxicology and Pharmaceutics (12)
	Dreyer, L. C. (3)	1994 (31)		
Tectonophysics (4)				
Neuroimage (3)				Veterinary (6)
Health Physics (3)		1993 (25)		
Naval Research Logistics (3)				Nursing (3)
American Society of Mechanical Engineers Pressure Vessels and Piping Division Publication PVP (3)		1992 (28)		Dentistry (3)
Natural Hazards (3)				
Hydrology and Earth System Sciences Discussions (3)				Arts and Humanities (2)
				Undefined (9)

Case studies are also commonly done in covering problems. Basic case studies in covering problems are shown in Table 2-2 below.

Table 2-2: Case Studies in Covering Problems (Farahani et al., 2012)

Paper description	Subject	Place
Current and O'Kelly (1992)	Emergency warning sirens	A mid western city
Osleeb and McLafferty (1992)	Facilities to fight Dracunculiasis (Guinea worm disease)	The Zou province, Benin
Repede and Bernardo (1994)	Emergency medical vehicles	Louisville, Kentucky
Ceria et al. (1998) and Caprara et al. (1999)	Crew scheduling	The Italian Railways Company
Murray (2005)	Emergency warning siren.	Dublin, Ohio
Farahani and Asgari (2007)	Locating military warehouses as distribution centers	Persia
Berman, Kalcsics, Krass, et al. (2009)	Specialized teams like police and firefighter	Gasoline transportation in Quebec and Ontario
Alexandris and Giannikos (2010)	Locating bank branches	Greece, Athens
Curtin et al. (2010)	Designing police patrol area	Dalas, Tx
Erdemir et al. (2010)	Locating aero medical and ground ambulances	USA, State of New Mexico
Lee and Murray (2010)	Locating Wi-Fi equipment	Dublin, Ohio
Oztekin et al. (2010)	An RFID network design methodology for asset tracking in healthcare	Stillwater Medical Center
Straitiff and Cromley (2010)	Location Set covering problem	Dublin, Ohio
Bell et al. (2011)	Locating air craft alert sites	USA

2.2 Set Covering Problem (SCP)

The set covering problem (SCP) is covering the rows of an m-row, n-column, zero-one matrix (a_{ij}) by a subset of the columns at minimum cost. Defining $x_j = 1$ if column j (with cost $c_j > O$) is in the solution and $x_j = 0$ otherwise (Beasley et al., 1996). The SCP model is;

Sets:

i: the index of demand nodes,

j: the index of facilities,

Parameters:

S: the maximum service distance,

cj: the cost of locating facility at node j and

aij: a binary parameter is 1 if distance from candidate place j to the existing facility i is not greater than S.

Decision Variable:

xj: a binary variable indicating whether the facility located at point j or not,

Mathematical Model:

Minimize $\sum_{j=1}^{n} c_j x_j$ (1)

Subject to

$\sum_{j=1}^{n} a_{ij} x_j \geq 1$ i= 1, ... m (2)

$x_j \in \{0,1\}$ j= l , ... n (3)

The objective function (1) minimizes total cost of the selected facilities. The constraints of equation (2) provide that each demand node i must be covered by at least one facility. And finally, the constraints (3) explain the domain of the decision

variables.

There are two main categories of set covering models in the literature. Location set covering problem (LSCP) is actually described by Church and ReVelle, with the purpose of covering all the customers to minimize the number of located facilities and maximal covering location problem (MCLP) is determined by ReVelle et al., with the purpose of maximizing total weighted covered customers according to specified number of facilities (Carlıoğlu, 2005).

The set covering problems and maximal covering problems are accepted as NP-hard in the literature (Karaman, 2008).

The SCP consists of finding a minimum cost set of the facilities among a finite set of candidate facilities so that all demand nodes are covered by at least one facility.

This study is actually based on maximal covering location problem (MCLP). Church and ReVelle (1974) improved a dual approach to set covering model. They offer a linear programming (LP) model maximizes coverage of total weighted demand with fixed number of facilities. The dual problem is called the Maximal Covering Location Problem (Töreyen, 2007).

To date, scarce application of the MCLP to locate ships for maximal covering has encountered in the literature. In the context of this study, it is shown that covering models can result in savings in terms of cost of operations and reduced response times.

Summary of deterministic set covering models and features of them are shown in Table 2-3 below.

Table 2-3: Deterministic Set Covering Models (Karaman, 2008)

Reference	Model	Objective	Coverage constraints	Constraints on location sites	Ambulances
ReVelle, Toregas, Bergman (1971)	LSCM	Minimize # of ambulances	Cover each demand point at least once	At most one ambulance per site	One type, number unlimited
Church and ReVelle(1974)	MCLP	Maximize the demand covered	None	At most one ambulance per site	One type, number given
Schilling et al. (1979)	TEAM	Maximize the demand covered	None	At most one ambulance of each type per site. Type A can only be located if Type B is located	Two types, number given
Schilling et al. (1979)	FLEET	Maximize the demand covered	None	At most one ambulance per site. Only p sites can be used	Two types, number given
Daskin and Stern (1981)	Modified MCLP	Maximize the demand covered, then the number of demand points covered more than once	Cover each demand point at least once	At most one ambulance per site	One type, number given
Hogan and ReVelle (1986)	Modified MCLP (BACOP1 and BACOP2)	Maximize the demand covered twice, or combination of the demand covered once or twice	Cover each demand point at least once	At most one ambulance per site	One type, number given
Gendreau et al. (1997)	DSM	Maximize the demand covered at least twice within r_1	All demand covered within r_2. Proportion α of all demand covered within r_1	Upper bound on the number of ambulances per site	One type, number given
Gendreau et al. (2001)	DDSM	Dynamically maximize the demand covered at least twice within r_1, minus a redeployment penalty term	All demand covered within r_2. Proportion α of all demand covered within r_1	Upper bound on the number of ambulances per site	One type, number given

Summary of probabilistic set covering models and features of them are shown in Table 2-4 below

Table 2-4: Probabilistic Set Covering Models (Karaman, 2008)

Reference	Model	Objective	Coverage constraints	Constraints on location sites	Ambulances	Busy Period
Daskin (1983)	MEXCLP	Maximize the expected demand covered	None	None	One type Upper bound given, (always reached)	Same for each ambulance, given
ReVelle and Hogan (1989)	MALP I	Maximize the total demand covered with a probability α	None	None	One type Number given	Same for all potential sites
ReVelle and Hogan (1989)	MALP II	Maximize the total demand covered with a probability at least α	None	None	One type Number given	Varies according to each demand point
Batta et al. (1989)	AMEXCLP	Maximize the expected demand covered	None	None	One type Number given	Varies according to each demand point
Ball and Lin (1993)	Modified LSCM (Rel-p)	Minimize the sum of ambulance fix costs	Proportion α of all demand covered within r_1	At most p_j ambulances at site j	One type Number given	Upper bound computed on busy period
Repede and Bernardo (1994)	TIMEXCLP	Maximize the expected demand covered	None	None	One type Number given Varying speeds	Same for each ambulance, given
Marianov and ReVelle (1994)	QPLSCP	Maximize the total demand covered with a probability at least α	None	None	One type Lower bound computed for each demand point	Varies according to demand points
Mandell (1998)	TTM	Maximize the expected demand	None	Bounds on each type per site	Two Types	Computed using a queueing model

2.3 Maximal Covering Location Problem (MCLP)

The maximal covering location problem (MCLP) has shown to be one of the most useful facility location models from both practical and theoretical points of view (Silva et al., 2001). The aim of MCLP is to construct a set of m facilities so as to maximize the total weight of covered customers, where a customer is considered as covered if it is located at most specific distance r away from the nearest facility (Berman et al., 2002). The maximal covering location model does not request that all demand points be covered.

The maximal covering location problem looks for the maximum population which can be served within a service distance or time given a limited number of facilities (Karasakal et al., 2004).

MCLP is mathematically formulated by Church and ReVelle (1974) as follows:

Sets:

I : set of demand nodes

J : set of facility sites

Parameters:

S : critical distance

d_{ij} : shortest distance from node *i* to node *j*

N : the set of facility sites that are eligible to cover demand point *i*,

$N_i = \{j \in J \mid d_{ij} \leq S\}$

a_i : population at demand node i

p : number of facilities

Decision Variables:

x_j : 1, if a facility is located at site *j*
0, otherwise

y_i : 1, if demand at *i* is covered
0, otherwise

Mathematical Model:

Maximize $\sum_{i \epsilon I} a_i y_i$ (1)

Subject to

$\sum_{j \epsilon N\, i} x_j \geq y_i \quad \forall i \in I$ (2)

$\sum_{j \epsilon J} x_j = p$ (3)

$x_j , y_i \in \{0,1\} \quad \forall i \in I , \forall j \in J$ (4)

Objective (1) maximizes the number of demand covered within critical distance. Constraint set (2) allows coverage of demand point *i* if one or more facilities are built within the critical distance. Constraint (3) limits the number of facilities built to *p*. Constraint set (4) ensures all variables to be binary.

The belief that mathematical location modeling can specify optimal location patterns rest on the basis that some exact objective can be identified. For example, in the area of private facilities location analysis, a fairly accurate statement of the objective of locating warehouses is to minimize the costs of manufacturing and distribution. Since most cost elements included in the objectives of private facility location can be reasonably guessed, the models can picture with some degree of certainty the real location problem they are designed to solve (Church and ReVelle, 1974).

The objectives of public facility location analysis are difficult to adopt and to quantify. The difficulty in defining direct measures of public aims has resulted in a

search for some substitute measure with which the decision maker may be comfortable. Two different substitute measures which have received attention in location models are:

- Total weighted distance for travel to the facilities.
- The distance that the user most distant from a facility would have to travel to achieve that facility, that is, the maximal service distance.

The use of a maximal service distance as a measure of the value of a given location mode has been discussed at length by Toregas and ReVelle who show that it is an significant substitute measurement for the value of a given location mode. For a given location solution, the maximum distance which any user would have to travel to achieve a facility would reflect the worst possible performance of the system. In the regional position of emergency facilities such as fire stations or ambulance dispatching stations, the concept of the maximal service distance is well established (Church and ReVelle, 1974).

The maximal service distance idea appears in the Location Set Covering Problem. This problem shows the minimal number and the location of facilities, which ensures no demand point will be farther than the maximal service distance from a facility. By solving the location set covering problem over a range of values of distance, it is possible to improve a cost-effectiveness curve from the pairs of numbers.

The number of facilities is used in the location set covering problem as the only cost element that enters the decision process. There exists evidence that insulating the number of facilities as the most special cost parameter may be valid in many real formulations.

Specifically, if unit costs are independent of site and number of demands covered, the use of number of facilities is a correct sign of cost. Examination of the cost-effectiveness curve discloses that for a given number of facilities there may be many location solutions which fulfill the requirement of coverage. For a desired level of expenditure then one may want to determine the solution with the smallest maximal service distance.

It could happen that the distance value obtained in this way is much larger than the desired distance S. If it is, the decision maker may shift the attention to concern with the total population covered within S. Having realized that his resources are insufficient to reach total coverage within his distance aim, the decision maker may look for to cover as many as possible within S using those limited resources. That is, facing the reality of an insufficient number of facilities, he may relinquish his aim of total coverage within S and attempt instead to locate the facilities in such a way that as few people as possible lie outside the wanted service distance (Church and ReVelle, 1974).

Some assumptions for MCLPs are as follows:

1. Various demand nodes may have various quantities of demand.

2. All facilities have a covering distance and only demand points within the covering distance are fulfilled.

3. There is no capacity limitation on facilities.

4. The number of facilities to be located is prearranged and the purpose is to maximize the demand covered.

Because the MCLP is shown to be extremely complex, some heuristic solution procedures have been developed. Additionally, the MCLP can be seen as a variant

formulation of other prominent location models.

Variations of the MCLP are formulated to include workload capacities or to maximize coverage and minimize distances to demands outside the maximum covering distance (Curtin et al., 2007).

For the LSCP and MCLP, if the cover distances are properly improved, then a population being covered in a solution to these types of models should mention that the population has good access to at least 1 facility (Ratick et al., 2009)

Among many different versions of MCLP models that is proposed, a basic assumption is that the facilities to be sited are uncapacitated. Under this assumption, the request will be served as long as it is within the service covering standard of any facility. However, this assumption of uncapacitated facilities severely limits the practice of covering models. Many service facilities have finite capacities to provide an acceptable level of service and spatial equity. For example, an ambulance base can only respond to a limited number of demands within its service covering standard at one time because of the availability condition of the ambulances stationed at the base (Yin et al., 2012).

The solution of maximal covering location problem is approached by a number of methods, both heuristic and exact. The approach used is depended on the context, apparently because the problem is rediscovered so many times. Church and ReVelle offered three methods for the problem. These include the greedy adding method, greedy adding and relaxed linear programming followed by branch and bound as needed. Galvao and ReVelle apply lagrangian relaxation to the this problem (ReVelle et al., 2008).

The solution of the large instances of MCLP is cumbersome, using exact methods. Therefore, heuristics have been employed to solve MCLPs of larger sizes. In Table 2-5, some heuristics for various types of MCLP are shown.

Table 2-5: Some Heuristics for MCLP (Zarandi et al., 2011)

Author(s)	Problem	Year
Galvão et al. [12]	MCLP	2000
Aytug and Saydam [13]	Maximum expected coverage	2002
Espejo et al. [14]	Hierarchical MCLP	2003
Park and Ryu [15]	Large-scale MCLP	2004
Karasakal and Karasakal [16]	Partial covering	2004
Shavandi and Mahlooji [17]	Fuzzy queuing maximal covering location	2006
Rajagopalan et al. [18]	Multi-period set covering problem	2008
ReVelle et al. [2]	MCLP	2008
Qu and Weng [19]	Hub-maximal covering location problem	2009
Canbolat and von Massow [20]	MCLP on the plane using ellipses	2009
Corrêa et al. [21]	Probabilistic covering location–allocation problem	2009
Lee and Lee [22]	Hierarchical MCLP	2010
Davari et al. [23]	MCLP with fuzzy travel times	2011
Basar et al. [24]	Multi-period double coverage	2011

The MCLP is proven to be nondeterministic polynomial time NP-hard (Yin, 2012), which means that no algorithm is discovered to solve it in polynomial time in the worst case.

2.4 Genetic Algorithm (GA)

Genetic Algorithm (GA) is used firstly by Holland in 1975 and aims to find a result that may be close to the best solution. This is the logic of natural selection, survival of the individual and powerful high probability of survival of an individual. In this method, the best results are taken as the result of evolution of individuals surviving. GA is a heuristic method (Altay, 2007).

Individuals in nature are combined to create children and these children are fundamental for survival of the fittest GA. GA begins with a community of individuals and gets the kids by these individuals. The number and the characteristics of individuals are randomly determined. These individuals are randomly matched and become new parents. Obtained from the children of this parent are uncovered to selection, crossover and exchange operations. The evolution is performed by repeating these ways. Completion of genetic algorithm depends on the finishing criteria.

In Figure 2-1, similarity between a numerical GA and biological genetics is shown.

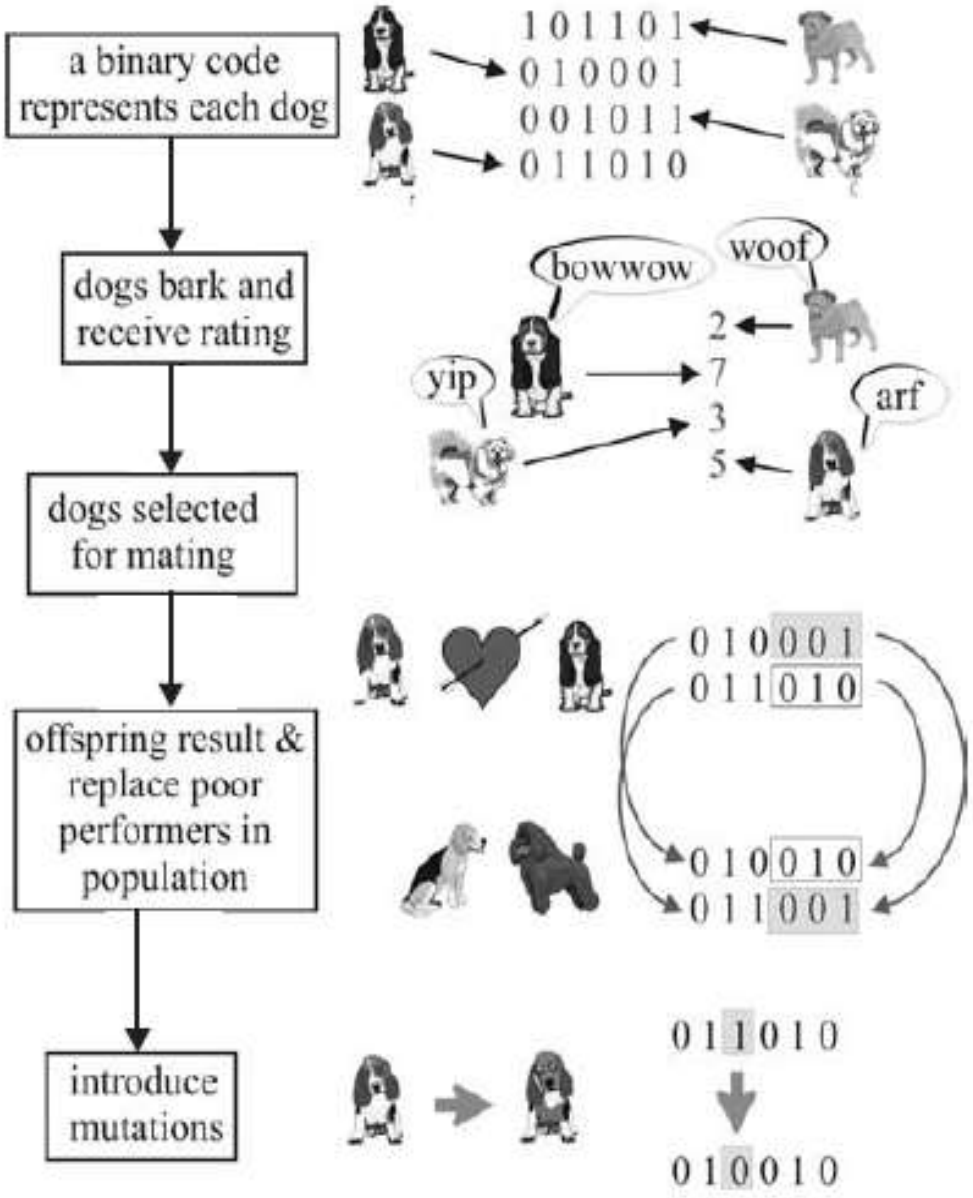

Figure 2-1: Similarity between a Numerical GA and Biological Genetics (Haupt et al., 2004)

The idea of GA is based on the evolutionary operation of biological organisms in nature. During the evolution, natural populations evolve according to the principles of selection and survival of the fittest. Individuals which are more successful in adapting to their environment have a better chance of surviving and reproducing, whilst individuals less fit will be eliminated. This means that the *genes*

from the highly fit individuals will spread to a rising number of individuals in each successive generation. The combination of good characteristics from adapted ancestors may produce even more fit offspring.

A GA simulates these operations by taking an initial population of individuals and applying genetic operators in all reproductions. In optimization terms, each individual in the population is encoded into a string or *chromosome* that represents a possible *solution* to a given problem. The fitness of an individual is measured with respect to a given objective function. Highly fit individuals are given opportunities to reproduce by exchanging pieces of their genetic information, in a *crossover* procedure, with other fit individuals.

This produces new offsprings which share some characteristics taken from both parents. Mutation is often applied after crossover by changing some genes in the strings. The offspring can either change the whole population or change less fit individuals. This evaluation cycle is repeated till a satisfactory solution is found (Beasley, 1996).

In Table 2-6, comparison of natural evolution and genetic algorithm terminology is shown.

Table 2-6: Comparison of Natural Evolution and Genetic Algorithm Terminology

(Haupt et al., 2004)

Natural evolution	Genetic algorithm
Chromosome	String
Gene	Feature or character
Allele	Feature value
Locus	String position
Genotype	Structure or coded string
Phenotype	Parameter set, a decoded structure

- **A Simple Genetic Algorithm**

A genetic algorithm is a problem solving method which uses genetics as its model of problem solving. It is a search method to find approximate solutions to optimization and search problems. Basically, an optimization problem works really simple. One knows the form of all the possible solutions corresponding to a specific question. The set of all solutions that meet this form constitute the search space. The problem consists in finding out the solution which fits the best, i.e. the one with the most payoffs, from all the possible solutions. If it is possible to quickly enumerate all solutions, the problem does not raise much difficulty. But, when the search space becomes large, enumeration is no longer feasible simply because it would take too much time. In this it is needed to use a specific technique to find optimal solution.

Genetic Algorithms ensures one of these methods. Practically they work in a similar way, adapting the simple genetics to algorithmic mechanisms. GA handles a population of all possible solutions. Each solution is represented through a chromosome that is just an abstract representation. Coding all possible solutions into a chromosome is the first part, but certainly not the most straightforward one of a

Genetic Algorithm. A set of reproduction operators is to be determined, too. Reproduction operators are applied directly on chromosomes, and are used to perform mutations and recombinations over solutions of the problem. Appropriate representation and reproduction operators are something determinant, as the behavior of the GA is extremely dependant on it.

Frequently, it can be difficult to find a representation that respects the structure of the search space and reproduction operators that are coherent and relevant according to the properties of the problems.

Selection is supposed to be able to compare all individuals in the population. Selection is made by using a fitness function. Each chromosome has an associated value corresponding to the fitness of solution it represents. The fitness should correspond to an evaluation of how good candidate solution is. Optimal solution is the one which maximizes the fitness function. Genetic Algorithms deal with the problems which maximize the fitness function. But, if the problem consists in minimizing a cost function, the adaptation is really easy. Either the cost function can be transformed into a fitness function, for example by inverting it; or the selection can be adapted in such way that they think individuals with low evaluation functions as better.

Once reproduction and fitness function have been properly defined, a Genetic Algorithm is evolved according to same basic structure. It begins by generating an initial population of chromosomes. This first population offers a wide diversity of genetic materials. The gene pool should be as large as possible so that any solution of search space can be engendered. Generally initial population is generated randomly. Then, genetic algorithm loops over an iteration operation to make the population

evolve (Sivanandam et al., 2008).

Each iteration consists of following steps:

• SELECTION: The first step consists in selecting individuals for the reproduction. This selection is done randomly with a probability depending on relative fitness of the individuals so that best ones are often chosen for reproduction than poor ones.

• REPRODUCTION: In the second step, offspring are bred by selected individuals. For generating new chromosomes, algorithm can use both recombination and mutation.

• EVALUATION: Then the fitness of new chromosomes is evaluated.

• REPLACEMENT: During the last step, individuals from the old population are killed and replaced by new ones. The algorithms stopped when population converges toward the optimal solution.

In Figure 2-2, the flowchart of a binary GA is shown.

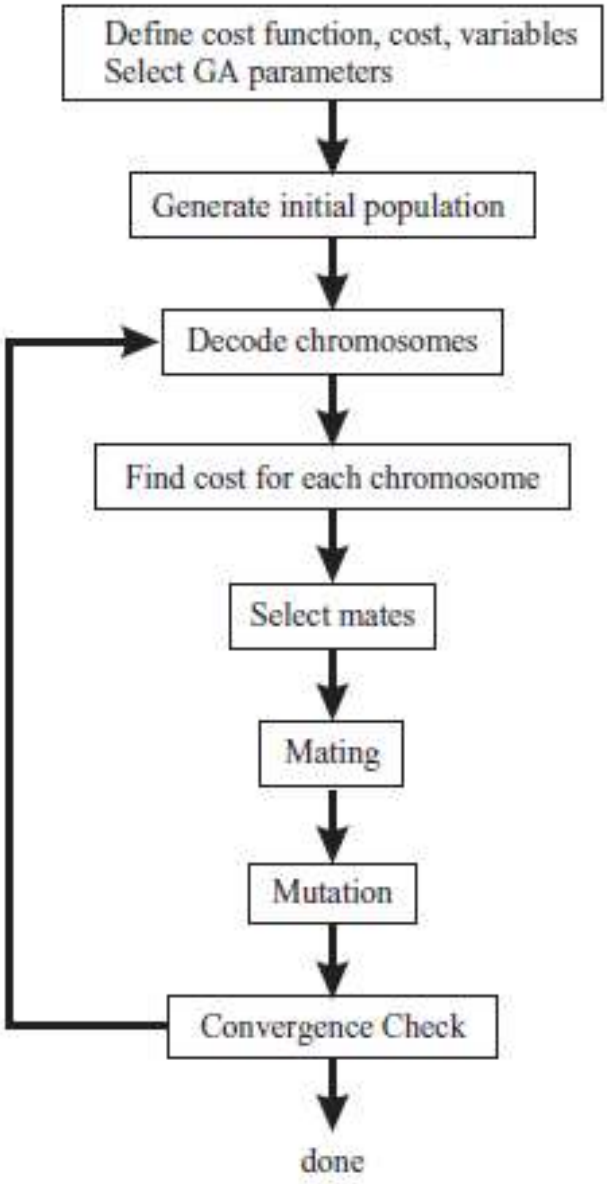

Figure 2-2: The Flowchart of a Binary Genetic Algorithm (Haupt et al., 2004)

In short, basic four steps used in simple Genetic Algorithm to solve a problem are,

1. Representation of the problem.

2. Fitness calculation.

3. Different variables and parameters involved in controlling the algorithm.

4. Representation of result and the way of terminating the algorithm.

From the mid 70s to date, various types of GA is employed in problems such as vehicle routing problems, portfolio selection, facility location, scheduling and quadratic assignment problems. The main thrust of GA is to improve generations gradually by using reproduction mechanisms, such as crossover and mutation. Each generation consists of a set of individual solutions and each iteration involves the selection of a set of parents, based on their fitness value and application of reproduction schemes to generate a set of offspring (Zarandi et al., 2011).

In Figure 2-3, general cycle of genetic algorithm is shown.

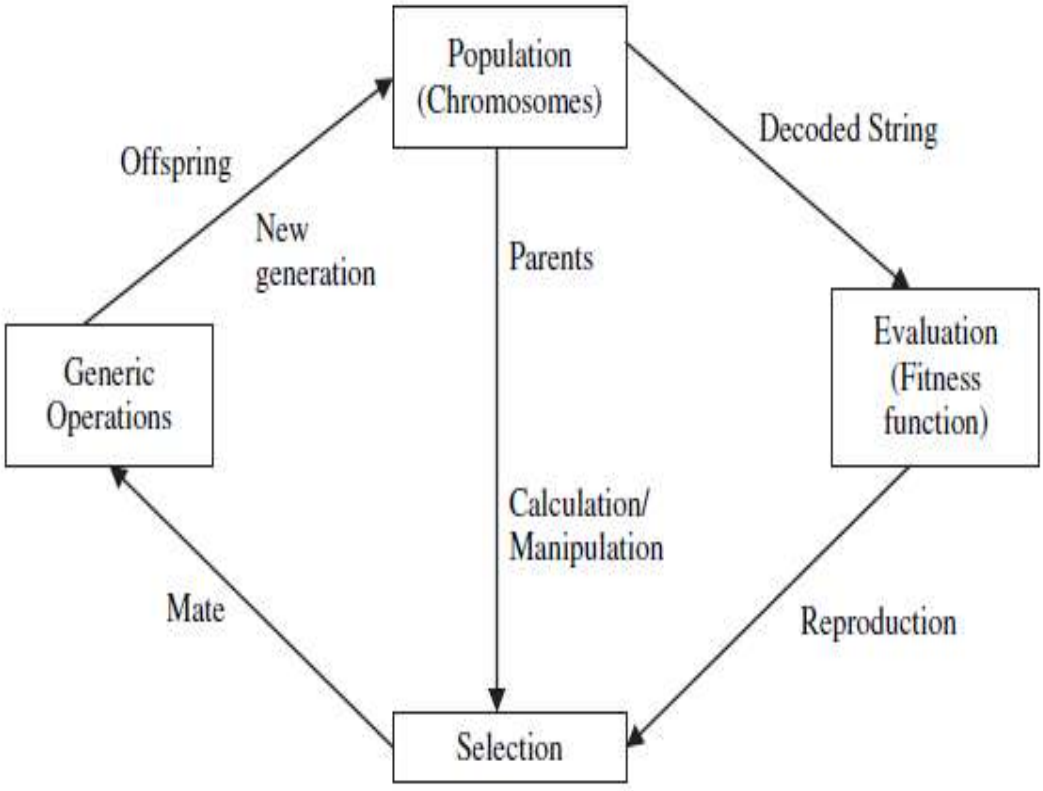

Figure 2-3: Genetic Algorithm Cycle (Sivanandam et al., 2008)

- **Advantages and Limitations of Genetic Algorithm**

The advantages of genetic algorithm include (Sivanandam et al., 2008),

1. Parallelism.
2. Liability.
3. Solution space is wider.
4. The fitness landscape is complex.
5. Easy to discover global optimum.
6. The problem has multi objective function.
7. Only uses function evaluations.
8. Easily modified for different problems.
9. Handles noisy functions well.
10. Handles large, poorly understood search spaces easily.
11. Good for multi-modal problems Returns a suite of solutions.
12. Very robust to difficulties in the evaluation of objective function.
13. They require no knowledge or gradient information about the response surface.
14. Discontinuities present on response surface have little effect on overall optimization performance.
15. They perform very well for large-scale optimization problems.
16. Can be employed for a wide variety of optimization problems.

The limitation of genetic algorithm includes,

1. The problem of identifying fitness function.
2. Definition of representation for the problem.

3. Premature convergence occurs.

4. The problem of choosing the various parameters like the size of the population, mutation rate, cross over rate, the selection method and its strength.

5. Cannot use gradients.

6. Cannot easily incorporate problem specific information.

7. Not good at identifying local optima.

8. No effective terminator.

9. Needs to be coupled with a local search technique.

10. Have trouble finding the exact global optimum.

11. Require large number of response function evaluations.

12. Configuration is not straightforward.

- **Applications of Genetic Algorithm**

Genetic algorithms is used for difficult problems (such as NP-hard problems), for machine learning and also for evolving simple programs. They are also used for some art, for evolving pictures and music. A few applications of GA are as follows (Sivanandam et al., 2008):

• Data analysis

• Robot trajectory planning

• Evolving LISP programs (genetic programming)

• Strategy planning

• Finding shape of protein molecules

• TSP and sequence scheduling

• Functions for creating the images

• Pole balancing, missile evasion, pursuit

• Aircraft design, keyboard configuration, communication networks

• Scheduling, facility scheduling, resource allocation

• Designing neural networks, both architecture and weights, improving classification algorithms, classifier systems

• Signal Processing

• Combinatorial optimization, set covering, traveling salesman (TSP), Sequence scheduling, routing, bin packing, graph coloring and partitioning.

- **Comparison of Genetic Algorithm Optimization and The Others**

Genetic algorithm differs from conventional optimization techniques in the following ways (Sivanandam et al., 2008):

1. Genetic algorithms operate with coded versions of problem parameters rather than parameters themselves, GA works with the coding of the solution set and not with the solution itself.

2. Almost all conventional optimization techniques search from a single point but GA always operates on a whole population of points. GAs use population of solutions rather than a single solution for searching. This plays a basic role to the robustness of genetic algorithms. It improves the chance of reaching global optimum and also helps in avoiding local stationary point.

3. Genetic Algorithm uses fitness function for evaluation rather than the derivatives. As a result, they can be applied to the any kind of continuous or discrete optimization

problem. The key point to be performed is to identify and specify a meaningful decoding function.

4. Genetic algorithms use probabilistic transition operates while conventional methods for continuous optimization use deterministic transition operates; GA does not use deterministic rules.

3 Definition and Mathematical Modeling of the Problem

Progress of maximal covering location problem and relations between the problem that is worked on in this study and original problem are handled in the following chapter. By explaining the missions and the features of interceptor baywatch vessels, it is expressed how they used in the mathematical model. And, the mathematical model of the problem is defined at the end.

3.1 The Maximal Covering Location Problem

Maximal Covering Location Problem (MCLP) is one of the known models for facility location problems. Algorithms to optimally solve this kind of operational management problem often suffer from combinatorial explosion when the system size increases. In these cases, heuristic algorithms are the only viable alternative. A lot of work is spent to the study of heuristics for MCLP.

Decision makers often have a wish to find optimal solutions to the various

practical operational management problems they meet every day. However, there is a class of common problems that are extremely difficult to solve optimally. Such problems include flow-shop and job-shop scheduling, lot sizing, assembly line balancing, project scheduling, facility location and so on.

Although algorithms in order to optimally solve such problems exist, they usually suffer from combination explosion. These problems belong to a class of problems that are known as NP-hard problems, and in most cases, the time and computing resources required to solve such problems become incredibly huge in practical applications makes traditional solution techniques infeasible. In these cases, heuristic methods are the only alternative (Xia et al., 2009).

In some location problems, there is a maximal service distance for each facility. For example, students a school need walk to schools if their home is within one mile range to their school. But, the public transportation must be provided for the students far away from one mile to their school. Another example is in the banking industry, bankers should ensure their service to the customers within a pre-determined distance in order to keep competitive with their adversaries. In the former case, the government may want to locate schools to minimize the number of students who must be bussed with public expense.

In the latter example, a banker may want to design its bank branches network to maximize number of potential customers within 500 meters to bank branches. In facility location literature, such a maximal service distance is known as covering distance. Demand within the covering distance is considered covered. An underlying assumption of this maximal distance is which demands are fully satisfied if there is a facility within the covering distance and are not satisfied otherwise.

On the other hand, in many facility planning situations, there exist also a budget constraint. For example, many bankers would like to have a service point within a short distance to all potential customers. However, satisfying such a requirement may need more service points and induce more cost of operation. Thus, it is needed to optimize the facility location in order to achieve a maximal covered customer demands subject to some budget constraints.

Location optimization problems under these assumptions are often called Maximal Covering Location Problems (MCLPs). It can be introduced briefly as follows. There are amount of predetermined facilities in a region. The service demands of customers are distributed in this area and each facility has a service range. All customers within a facility's service range are covered by this facility. The objective is to optimize the locations of all facilities for maximizing the total demands amount covered by these facilities (Xia et al., 2009).

In this study, the problem "maximal covering location of interceptor baywatch vessels" is dealt. The aim of this problem is to locate interceptor baywatch vessels for maximal covering of particular sea area. And when a new illegal event or a help call appears in the sea area, an interceptor baywatch vessel will certainly interfere in an hour.

For this purpose, a mathematical model is developed. Set covering models are examined and because vessel number is evident and the aim is to maximize the demand area, maximal covering location model is seen to be appropriate for this problem.

In addition to special features of MCLP model, unique constraints of this problem is integrated. In the model, certain number and qualification of ships are

intended to appoint certain pre-defined ports providing the prescribed constraints in order to minimize the cost.

3.2 Missions and Features of Interceptor Baywatch Vessels

While developing the mathematical model, missions and features of interceptor baywatch vessels are considered. In order to cover all the sea area, especially speed and draft values of the ships are very important.

In general, the mission of interceptor baywatch vessels is to enforce national and international laws and to ensure the safety of life and property within its area of maritime jurisdiction.

Missions of interceptor baywatch vessels which have responsibility and authority over the seas are as follows:

- To protect and provide the security in maritime jurisdiction area.
- To provide safety of life and property at sea.
- To take necessary measures for untethered mines, explosives and suspicious.
- To report the material identified in the sea and on coast to the authorities concerned.
- To observe and inspect the operating conditions of aids-to-navigation and report the deficiencies observed to the authorities concerned.
- To disarm the refugees entering into our territorial waters and deliver them to the authorities concerned.
- To prevent all kinds of smuggling carried out by way of sea.
- To prevent the actions of the vessels and sea craft in violation of the laws on

radio hygiene, passport, anchoring, mooring, fishing, diving and hoisting the flag.

- To inspect the fishing of aquatic products.
- To conduct inspections in order to prevent marine pollution.
- To prevent the smuggling on antiquities by conducting inspections on diving activities.
- To perform search and rescue missions within the search and rescue area, in conformity with the International Search and Rescue Convention and National Search and Rescue Regulations.
- To inspect yacht tourism,

The features of the interceptor baywatch vessels according to the classes are presented in the following tables.

Table 3-1: Class 6 Ships Basic Specifications

Dimensions (Length-Width-Draft)	34.60 M - 8.60 M - 3.0 M
Displacement Tonnage	180 Ton
Propulsion	3200 HPx2
Maximum Speed	17 KTS
Economic Speed / Range	15 KTS - 600 NM

Table 3-2: Class 7 Ships Basic Specifications

Dimensions (Length-Width-Draft)	36.60 M - 8.60 M - 3.0 M
Displacement Tonnage	180 Ton
Propulsion	3200 HPx2
Maximum Speed	30 KTS
Economic Speed / Range	22 KTS - 500 NM

Table 3-3: Class 8 Ships Basic Specifications

Dimensions (Length-Width-Draft)	40.75 M - 7.05 M - 2.0 M
Displacement Tonnage	156 - 195 Ton
Propulsion	2692 HPx2
Maximum Speed	27 KTS
Economic Speed / Range	20 KTS - 1000 NM

Table 3-4: Class 1 Ships Basic Specifications

Dimensions (Length-Width-Draft)	31.50 M - 6.7 M - 2.0 M
Displacement Tonnage	83 - 97 Ton
Propulsion	40000 HPx2
Maximum Speed	47 KTS
Economic Speed / Range	35 KTS - 500 NM

Table 3-5: Class 3 Ships Basic Specifications

Dimensions (Length-Width-Draft)	35.69 M - 6.7 M - 2.0 M
Displacement Tonnage	87 - 113 Ton
Propulsion	3700 HPx2
Maximum Speed	45 KTS
Economic Speed / Range	40 KTS - 500 NM

3.3 Mathematical Modeling of the Problem

In order to achieve this mission, a 0-1 integer programming model is defined. The assumptions, sets, parameters and mathematical model of the problem are given below.

Assumptions:

1. All the ships sail always with maximum speed.
2. All the ships begin sailing immediately without any preparation time.
3. All the ships are always full of fuel and never break down.
4. Safety drafts of all the ships are 1 meter more of own draft for secure location.

Sets:

I : set of ships $\in \{1,2,\ldots,n\}$

J : set of ports $\in \{1,2,\ldots,m\}$

Parameters:

N : total number of ships

m_{ij}: cost of assignment of ships to ports

b_j : depth of port j

d_i : safety draft of ship i

s_i : maximum speed of ship i

u_j : covering distance

t : reaction time for the emergency situation

Decision variables:

x_{ij} : $\begin{cases} 1, & \text{if ship i is assigned to port j} \\ 0, & \text{otherwise} \end{cases}$

y_{ij} : auxiliary binary variables for providing if-then structure

z_{ij} : auxiliary binary variables for providing if-then structure

Mathematical Model:

Minimize $\sum_{i=1}^{n} \sum_{j=1}^{m} m_{ij} x_{ij}$

Subject to

$$d_i x_{ij} - b_j \leq M y_{ij} \qquad \forall i \in I, \forall j \in J \qquad (1)$$

$$x_{ij} \leq M (1 - y_{ij}) \qquad \forall i \in I, \forall j \in J \qquad (2)$$

$$u_j - t s_i x_{ij} \leq M z_{ij} \qquad \forall i \in I, \forall j \in J \qquad (3)$$

$$x_{ij} \leq M (1 - z_{ij}) \qquad \forall i \in I, \forall j \in J \qquad (4)$$

$$\sum_{i=1}^{n} x_{ij} \geq 1 \qquad \forall j \in J \qquad (5)$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \forall i \in I \qquad (6)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} = N \qquad (7)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \qquad (8)$$

$$y_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \qquad (9)$$

$$z_{ij} \in \{0,1\} \qquad \forall i \in I, \forall j \in J \qquad (10)$$

Objective function minimizes the cost of ship assignment to a port. All the ships are assigned to ports by considering both features of the ships and location costs of the ports.

Constraint (1) and constraint (2) imply ship assignment to a port for suitable draft and depth values. Constraint (1) and constraint (2) ensure "**if** $x_{ij} > 0$ **then** $d_i x_{ij} \leq b_j$" constraint. If a ship assigns to a port then safety draft of the ship must be equal or less than depth of the port.

Constraint (3) and constraint (4) guarantee ship assignment to a port for suitable covering distance and speed values. Constraint (3) and constraint (4) ensure "**if** $x_{ij} > 0$ **then** $t\, s_i\, x_{ij} \geq u_j$" constraint. If a ship assigns to a port then speed of the ship must be equal or over the maximum covering distance of the port.

Constraint (5) ensures all ports to be assigned to at least one ship. Therefore, all ports are fulfilled.

Constraint (6) guarantees all ships to be assigned to only one port. So, all ships are in only one port.

Constraint (7) implies using certain number of ship because the ships on hand are pre-known.

Constraint (8), constraint (9) and constraint (10) are binary variables imposed to the model.

4 Solution Methodology

Optimal and heuristic approaches are presented in the following chapter. Solution approaches for the problem worked on in this study and how optimal and genetic algorithm solutions are applied on the problem are expressed.

4.1 Solution Approaches

Three solution approaches are available in the literature for location problems. One is the complete enumeration, which is easy to implement when problem size is small. Since every possible solution is signed in the complete enumeration, the computational load is high. The other solution approach is mathematical programming. The mathematical models guarantee the optimal solutions. Even with the aid of the advanced mathematical solvers they may not solve large-size problems. The other solution approach is heuristics that are capable of solving large-size

problems, but fail to provide optimality.

Several heuristics that are used for location problems are greedy adding, simulated annealing, tabu search, genetic algorithms and lagrangian relaxation based heuristics.

Upon reviewing the solution approaches and methods in the literature, genetic algorithm that is proved to be producing reliable results in location problems is used (Karaman, 2008).

Some of the advantages of a GA (Haupt et al., 2004):

• Optimizes with continuous or discrete variables,
• Does not request derivative information,
• Simultaneously searches from a wide sampling of the cost surface,
• Deals with a large number of variables,
• Is well suited for parallel computers,
• Optimizes variables with extremely complex cost surfaces,
• Provides a list of optimum variables, not just a single solution,
• May encode the variables so that the optimization is done with the encoded variables,
• Works with the numerically generated data, experimental data, or analytical functions.

These advantages are also the reason for using genetic algorithm. Of course, the GA is not only the best way to solve every problem. For instance, the traditional methods are tuned to quickly find the solution of a well behaved convex analytical function of only a few variables. For such cases the calculus-based methods

outperform the GA, quickly finding the minimum while the GA is still analyzing costs of the initial population. For these problems the optimizer uses the experience of the past and employs these quick methods. However, many realistic problems do not fall into this category. In addition, for problems which are not difficult, other methods may find the solution faster than the GA. (Haupt et al., 2004).

4.2 Optimal Solution

After developing the mathematical model, different data sets are generated for small size of problems and it is tried to find optimal solutions for them. Validation of the mathematical model is tested by changing parameters and sizes of the problems. And it is shown that for all conditions, the model is right and assignments are correct. Some of the optimal solution experiments are described in chapter 5.

Binary Integer Programming Tool in MATLAB Global Optimization Toolbox is used for optimal solutions.

For big sizes of problems, MATLAB cannot solve the problem optimally. Therefore, the problem is solved by using genetic algorithm.

4.3 Genetic Algorithm Solution

The genetic algorithm is a method for solving both unconstrained and constrained optimization problems which is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly changes a population of individual solutions. At each step, the genetic algorithm chooses

individuals at random from the current population to be parents and uses them to produce the children for next generation. Over successive generations, the population improves toward an optimal solution.

Genetic Algorithm Tool in MATLAB Global Optimization Toolbox is used for optimal solutions.

The pseudocode of the genetic algorithm is shown in Figure 4-1 (MATLAB, 2013) below:

Begin

time = 0;

Generate(constraint dependant) random initial Population(time);

Evaluate Population(time);

While stopping criterion not meet **Do**

Begin

time = time + 1;

Select(stochastic uniform) Population(time) from Population(time-1);

Crossover(scattered) pairs in Population(time);

Mutate(constraint dependant-adaptive feasible) offspring;

Evaluate Population(time);

End

Output the fittest individual;

End

Figure 4-1: Pseudocode of the Genetic Algorithm

The following statements summarize how the genetic algorithm works (MATLAB, 2013):

The algorithm starts by creating (constraint dependant) a random initial population. Then the algorithm generates a sequence of new populations. At each step, the algorithm uses the individuals to generate next population in the current generation. In order to generate new population, the algorithm makes the following steps:

- Scores all members by computing its fitness value.

- Scales the fitness scores to transform them into a proper range of value.

- Chooses members based on fitness values.

- Some individuals having lower fitness are chosen as elite. These ones are passed to next population.

- Generates children from parents. Children are generated by making random changes to a single parent (mutation) or by joining the vector entries of a couple of parents (crossover).

- Replaces the present population with the children in order to form the next generation.

- The algorithm halts when one of the stopping criteria is satisfied.

After generating a random initial population, at each step, the genetic algorithm uses the current population to generate the children that make up the next generation. The algorithm chooses a group of individuals in the current population, called parents, who contribute their genes to their children. The algorithm usually chooses

individuals that have better fitness values as parents.

In Figures 4-2, 4-3 and 4-4, representations of chromosome, genes and population are shown.

Figure 4-2: Chromosome

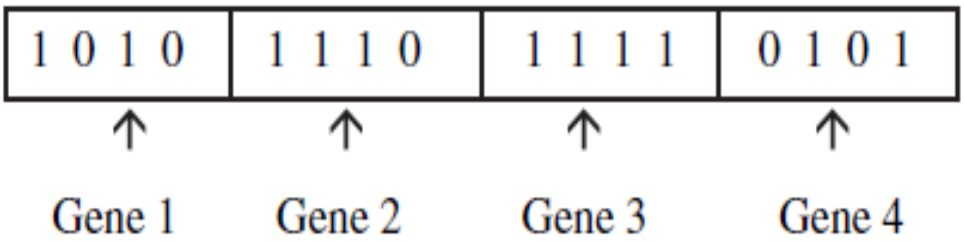

Figure 4-3: Genes

Population	Chromosome 1	1 1 1 0 0 0 1 0
	Chromosome 2	0 1 1 1 1 0 1 1
	Chromosome 3	1 0 1 0 1 0 1 0
	Chromosome 4	1 1 0 0 1 1 0 0

Figure 4-4: Population

The genetic algorithm generates three types of children for the next generation. Elite children are the individuals in the present generation with the best fitness values. These individuals automatically keep alive to the next generation. Crossover

children are generated by combining the vectors of a pair of parents. Mutation children are generated by making random changes, or mutations, to a single parent.

Mutation and Crossover expresses how to specify the number of children of each type that the algorithm produces and the functions it uses to perform crossover and mutation.

The fitness scaling function transforms the raw fitness scores returned by the fitness function to values in a range that is appropriate for the selection function. Scaling function indicates function that performs the scaling. Rank which is used in this algorithm as the scaling function scales the raw scores which is based on the rank of each individual, rather than its score. The rank of an individual is its location in the sorted scores. The rank of the fittest individual is 1, the next is 2, and so on. Rank fitness scaling clears the effect of spread of the raw scores.

The selection function chooses parents based on their scaled values from the fitness scaling function. Stochastic uniform selection which is used in this algorithm lays out a line in which each parent communicates to a section of the line. The algorithm acts along the line in steps of equal size, one step for each parent. At each step, the algorithm devotes a parent from the section it lands on. The first stage is a uniform random number less than the stage size. In Figure 4-5 (Robles, 2012), representation of a stochastic uniform selection is shown.

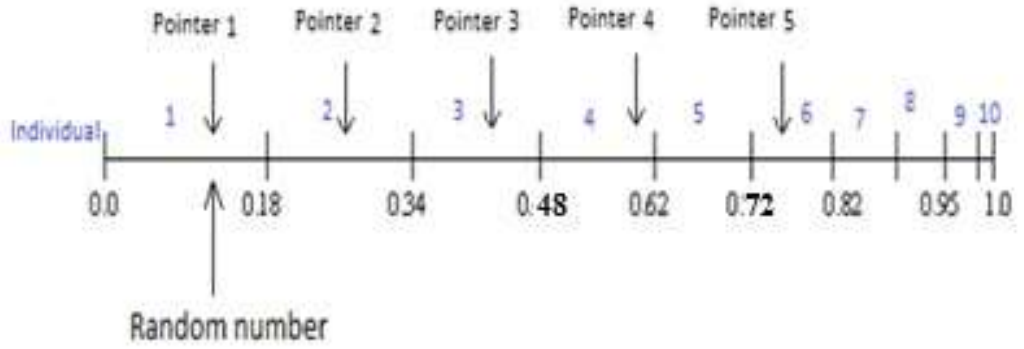

Figure 4-5: Stochastic Uniform Selection

Elite count states the number of individuals that are guaranteed to keep alive to the next generation.

Crossover joins two individuals to form a new individual for the next generation. Scattered crossover which is used in this algorithm creates a random binary vector. It selects the genes where vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent. Crossover fraction indicates the fraction of the next generation that crossover produces. Mutation produces the remaining individuals in the next generation. For each location in the gene, a random number decides if the feature comes from parent 1 or parent 2. In Figure 4-6 (Robles, 2012), an illustration is given, where if the random value for a given location is higher than 0.5 parent 2 is chosen, otherwise parent 1 is chosen.

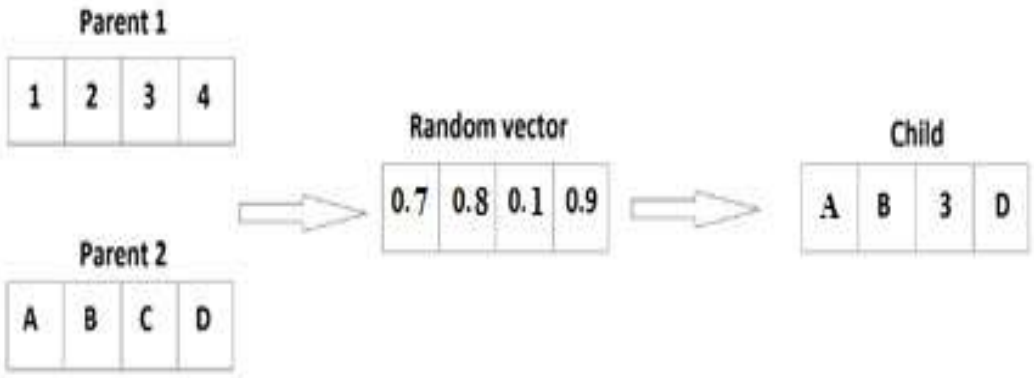

Figure 4-6: Scattered Crossover

Mutation function does random changes in the individuals in the population, which provide genetic diversity. Constraint dependant adaptive feasible mutation which is used in this algorithm randomly produces directions that are adaptive with respect to the last successful generation. In Figure 4-7, representation of a mutation is shown.

$$0010110000001 \Rightarrow 0010100000001$$

Figure 4-7: Mutation

Migration is the action of individuals between subpopulations which the algorithm generates. So often, the best individuals from one subpopulation change the worst individuals in another subpopulation. Direction to forward migration which is used in this algorithm occurs toward the last subpopulation. In Figure 4-8, a representation of forward migration is shown.

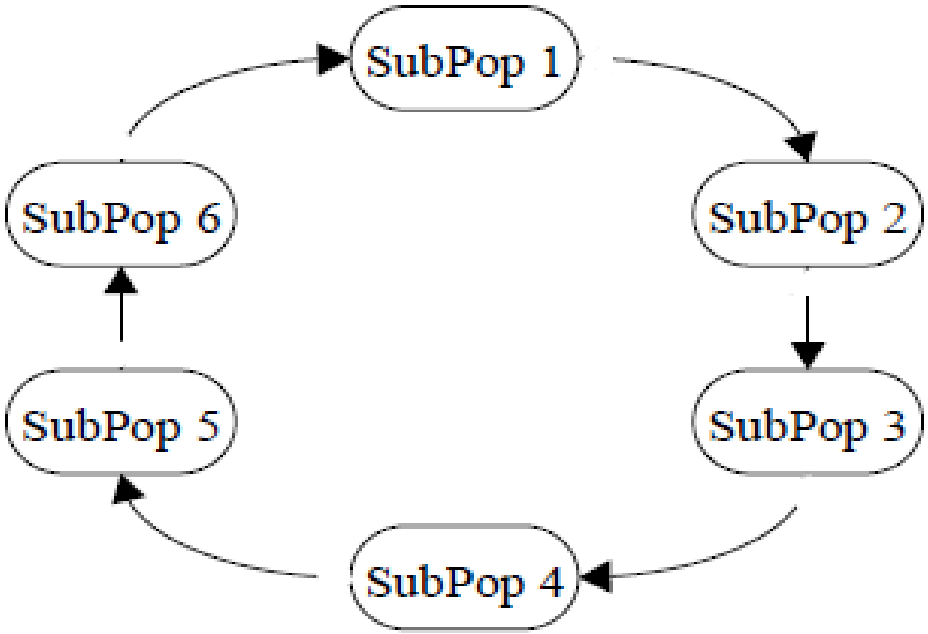

Figure 4-8: Forward Migration

Migration winds at the ends of the subpopulations. That means the last subpopulation migrates into the first, and the first one may migrate into the last.

Migration fraction checks how many individuals act between subpopulations. Fraction is the part of the smaller of the two subpopulations that acts.

5 Experimental Study

Experimental study which is done for the problem worked on in this thesis is explained. Solution methodologies expressed in chapter 4 are applied to the problem and experiments for coverage of Turkish seas are done. First parameters are explained and then coverage of all Turkish seas is shown. At the end, comparison of optimal and genetic algorithm solutions is expressed in the following chapter.

Binary Integer Programming Tool in MATLAB Global Optimization Toolbox is used for optimal solution. In Figure 5-1, the interface of Binary Integer Programming Tool is shown.

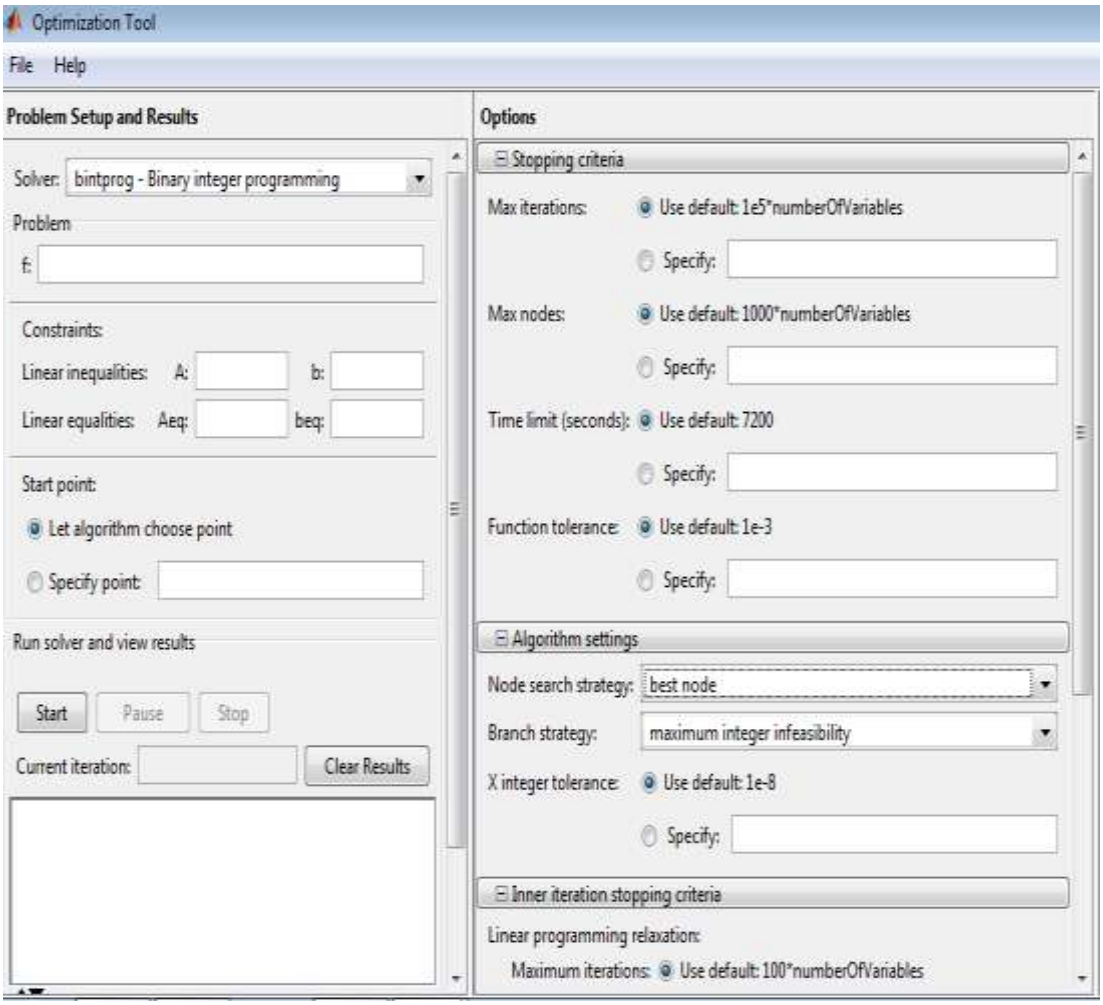

Figure 5-1: Binary Integer Programming Tool

Binary Integer Programming Tool parameters that are used for optimal solutions and explained in Appendix B exhaustively are shown in Table 5-1.

Table 5-1: Binary Integer Programming Tool Parameters

Max iterations	1e-5 * Number of Variables
Max nodes	1000 * Number of Variables
Time limit (seconds)	7200
Function tolerance	1e-3
Node search strategy	Best Node
Branch strategy	Maximum Integer Infeasibility
X integer tolerance	1e-8
Maximum iterations	100 * Number of Variables
Level of display	1e-6
Node interval	Off
Function tolerance	20

Default parameters of MATLAB Binary Integer Programming Tool values showed above are used for all experiments.

Genetic Algorithm Tool in MATLAB Global Optimization Toolbox is used for genetic algorithm solution. In Figure 5-2, the interface of Genetic Algorithm Tool is shown.

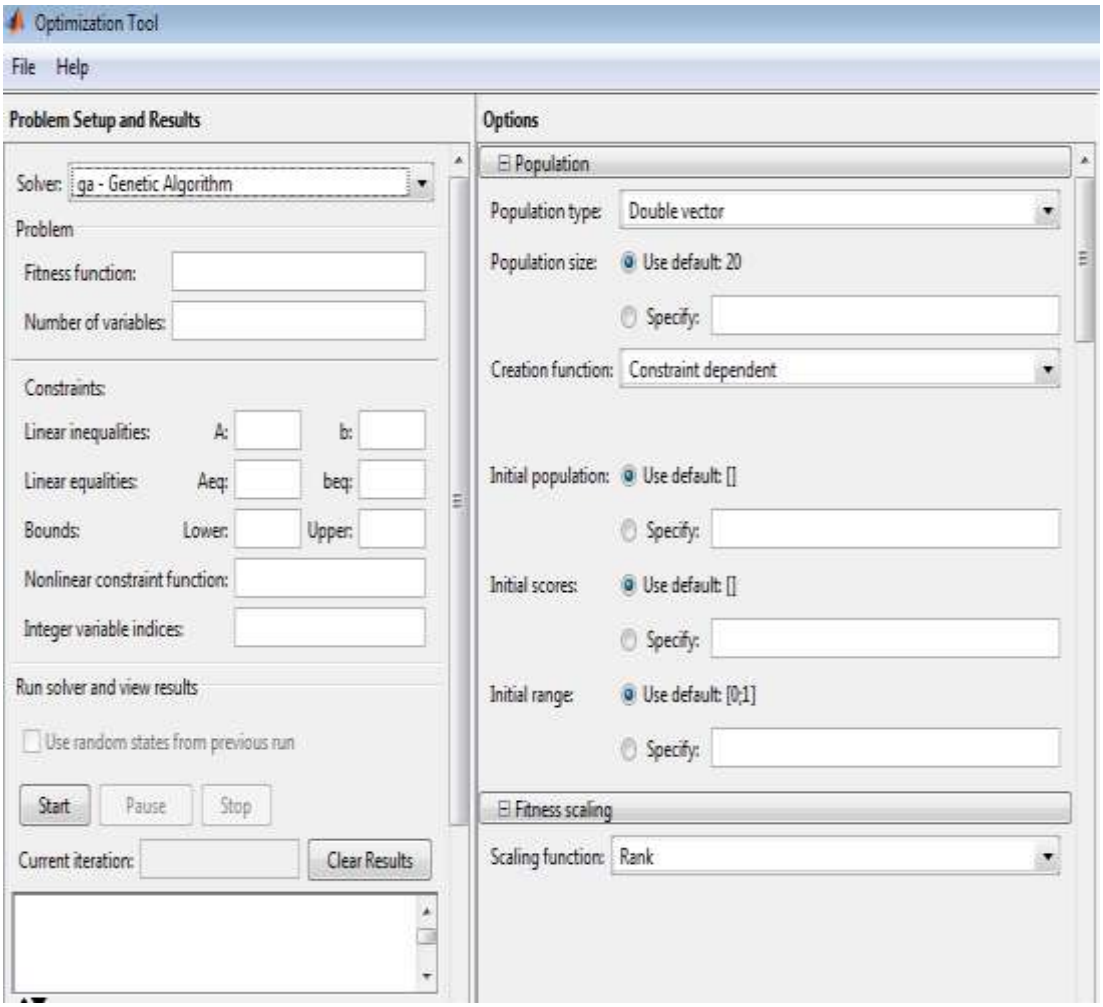

Figure 5-2: Genetic Algorithm Tool

Because of integer variable constraints, Genetic Algorithm Tool allows only a few changes on parameters. That's why only best crossover fraction value is attempted to find. And because fewer than 100 stall generations solver cannot satisfy constraints of the problem, stall generations value is set to 100. The other values are set to default. Best crossover fraction value is generally between 0.6 and 1.0 (Altay, 2007). Therefore, some experiments are done for finding the best crossover fraction value. And, 0.8 is found as best crossover fraction value. The crossover fraction value 0.8 is already default value of MATLAB.

All experiments are computed in computer with Intel N450 CPU 1.67 GHz and 2 GB RAM properties.

The experiments which are done for finding the best crossover fraction value are shown below in Table 5-2.

Table 5-2: Experiments for Crossover Fraction Value

Problem Size		**Best GA Solutions Obtained with Crossover Fraction Values**					**Optimal/Best Solution**
Number of Ships	**Number of Ports**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**	
5	3	23	23	**23**	23	24	23
6	3	25	24	**24**	25	25	24
7	3	27	27	**27**	27	28	26
8	3	29	28	**28**	28	29	27
6	4	28	28	**28**	28	29	27
7	4	28	28	**28**	29	29	28
8	4	33	33	**32**	32	33	32
9	4	35	34	**34**	35	35	33
7	5	31	32	**30**	31	32	29
8	5	35	35	**34**	34	35	33
9	5	38	38	**36**	38	38	35*
10	5	37	38	**37**	37	37	37*
15	10	61	61	**59**	61	61	57*

*Best solution after 2 hours of run.

Genetic Algorithm Tool parameters that are used for genetic algorithm solutions and explained in Appendix C exhaustively are is shown in Table 5-3 below:

Table 5-3: Parameters of Genetic Algorithm Tool

Population	Double vector
Population size	100
Creation function	Constraint dependant
Fitness scaling	Rank
Selection	Stochastic Uniform
Elite Count	2
Mutation	Constraint dependent – Adaptive feasible
Crossover	Scattered
Crossover Fraction	0.8
Migration	Forward Direction
Generations	100
Time limit	Infinitive
Fitness limit	Infinitive
Stall Generation	100
Stall time limit	Infinitive

5.1 Coverage of Black Sea

Coefficients of objective function and constraints which are used in experiments for coverage of Black Sea are shown in Table 5-4 below.

Table 5-4: Coefficients of Objective Function and Port Constraints

PORT NAME	LOCATION COST OF PORT(UNIT)	COVERAGE DISTANCE OF PORT(NM)	DEPTH OF PORT(M)
HOPA	3	12	3
RIZE	3	12	3
TRABZON	2	12	4
GIRESUN	3	12	3
FATSA	3	12	3
SAMSUN	2	12	5
SINOP	3	12	4
INEBOLU	3	12	3
AMASRA	2	12	4
KDZ. EREGLI	2	12	5

Coefficients of ship constraints which are used in experiments for coverage of Black Sea are shown in Table 5-5 below.

Table 5-5: Coefficients of Ship Constraints

CLASS OF SHIP	ID OF SHIP	SPEED OF SHIP(KTS)	SAFETY DRAFT OF SHIP(M)
6	63	17	4
6	64	17	4
6	69	17	4
7	71	30	4
7	72	30	4
8	86	27	3
8	87	27	3
8	95	27	3
8	96	27	3
8	97	27	3
3	301	45	3
3	307	45	3

The problem cannot be solved by MATLAB optimally under the reasonable time limit because of variable and constraint number. Best lower bound on objective value, which is 30, is accepted as best solution.

Assignments which are appeared after genetic algorithm solution are shown in Table 5-6 below.

Table 5-6: Ship and Port Assignments for Coverage of Black Sea

Cost= 30 Gap= 0 % Time:49 sec.	**63**	**64**	**69**	**71**	**72**	**86**	**87**	**95**	**96**	**97**	**301**	**307**
HOPA	0	0	0	0	0	0	1	0	0	0	0	0
RIZE	0	0	0	1	0	0	0	0	0	0	0	0
TRABZON	0	0	1	0	0	0	0	0	0	0	1	0
GIRESUN	0	0	0	0	1	0	0	0	0	0	0	0
FATSA	0	0	0	0	0	0	0	0	1	0	0	0
SAMSUN	1	0	0	0	0	0	0	0	0	0	0	0
SINOP	0	0	0	0	0	0	0	1	0	0	0	0
INEBOLU	0	0	0	0	0	0	0	0	0	0	0	1
AMASRA	0	1	0	0	0	1	0	0	0	0	0	0
KDZ. EREGLI	0	0	0	0	0	0	0	0	0	1	0	0

In Figure 5-3, ship and port assignments for coverage of Black Sea is shown on the map of Black Sea.

Figure 5-3: Ship and Port Assignments for Coverage of Black Sea

In Figure 5-4, graphical representation of best fitness and best individual results of genetic algorithm solution for coverage of Black Sea is shown.

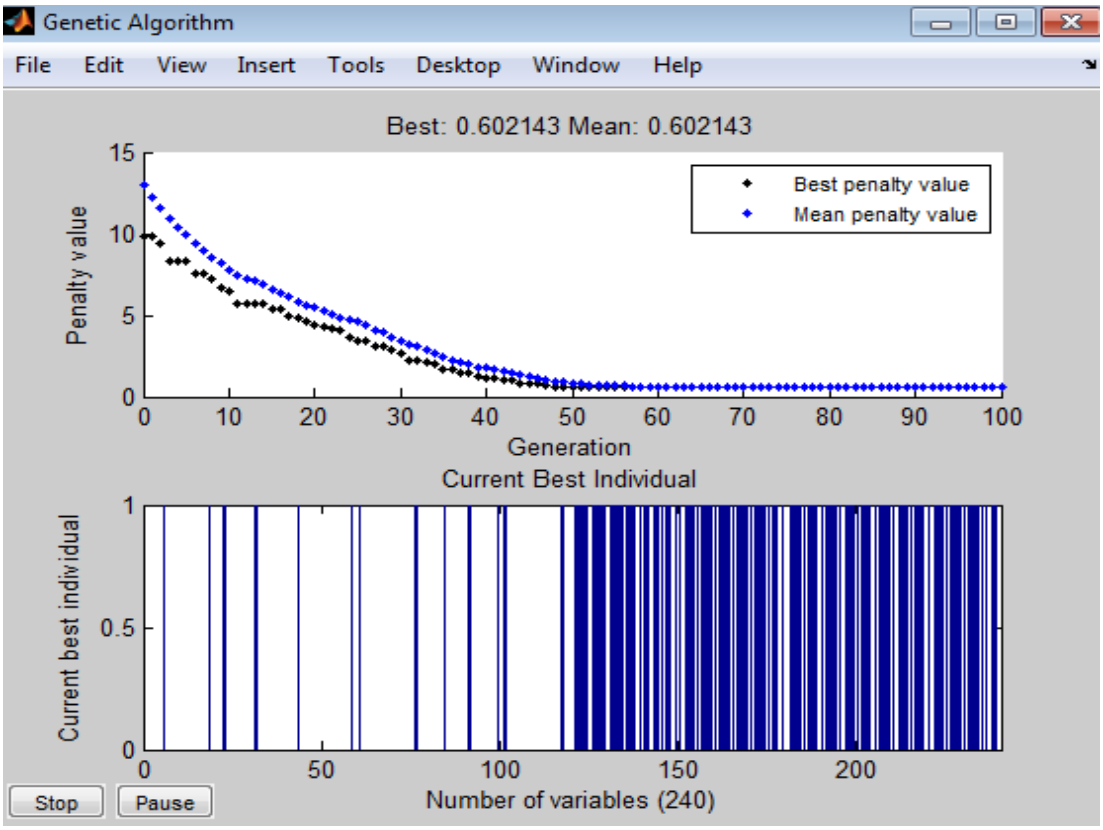

Figure 5-4: Graphical Representation of Best Fitness and Best Individual Results

In the first graphic, minimization of fitness function values plotted for each generation is shown. Nearly at the half of the all generations, minimization is almost done. In the second graphic, variables which are in the first half of the all variables represents the ships assigned to a port. The lines are 1s and the others are 0s. The second half of the all variables means nothing because they are auxiliary variables.

5.2 Coverage of Marmara Sea

Coefficients of objective function and constraints which are used in experiments for coverage of Marmara Sea are shown in Table 5-7 below.

Table 5-7: Coefficients of Objective Function and Port Constraints

PORT NAME	LOCATION COST OF PORT(UNIT)	COVERAGE DISTANCE OF PORT(NM)	DEPTH OF PORT(M)
KEFKEN	3	12	4
IGNEADA	3	12	4
ISTANBUL	2	12	5
MAR.EREGLISI	3	12	3
BANDIRMA	3	12	4
CANAKKALE	2	12	5
ENEZ	3	12	3

Coefficients of ship constraints which are used in experiments for coverage of Marmara Sea are shown in Table 5-8 below.

Table 5-8: Coefficients of Ship Constraints

CLASS OF SHIP	ID OF SHIP	SPEED OF SHIP(KTS)	SAFETY DRAFT OF SHIP(M)
6	65	17	4
6	66	17	4
7	74	30	4
8	84	27	3
8	85	27	3
8	88	27	3
8	90	27	3
8	91	27	3
8	93	27	3
8	94	27	3
1	106	47	3
3	303	45	3
3	308	45	3
3	312	45	3

The problem cannot be solved by MATLAB optimally under reasonable time limit because of variable and constraint number. Best lower bound on objective value, which is 33, is accepted as best solution.

Assignments which are appeared after genetic algorithm solution are shown in Table 5-9 below.

Table 5-9: Ship and Port Assignment for Coverage of Marmara Sea

Cost = 34 Gap= 3.03 % Time:41 sec.	**65**	**66**	**74**	**84**	**85**	**88**	**90**	**91**	**93**	**94**	**106**	**303**	**308**	**312**
KEFKEN	0	0	0	0	0	0	0	0	1	0	0	0	0	0
IGNEADA	0	0	0	1	0	0	0	0	0	0	0	0	0	0
ISTANBUL	0	0	1	0	0	1	1	0	0	0	0	1	0	0
MAR.EREGLI	0	0	0	0	0	0	0	0	0	1	0	0	0	0
BANDIRMA	0	0	0	0	0	0	0	0	0	0	1	0	0	1
CANAKKALE	1	1	0	0	1	0	0	0	0	0	0	0	0	0
ENEZ	0	0	0	0	0	0	0	1	0	0	0	0	1	0

In Figure 5-5, ship and port assignments for coverage of Marmara Sea is shown on the map of Marmara Sea.

In Figure 5-6, graphical representation of best fitness and best individual results of genetic algorithm solution for coverage of Marmara Sea is shown.

In the first graphic, minimization of fitness function values plotted for each generation is shown. Nearly at the half of the all generations, minimization is almost done. In the second graphic, variables which are in the first half of the all variables represents the ships assigned to a port. The lines are 1s and the others are 0s. The second half of the all variables means nothing because they are auxiliary variables.

Figure 5-5: Ship and Port Assignments for Coverage of Marmara Sea

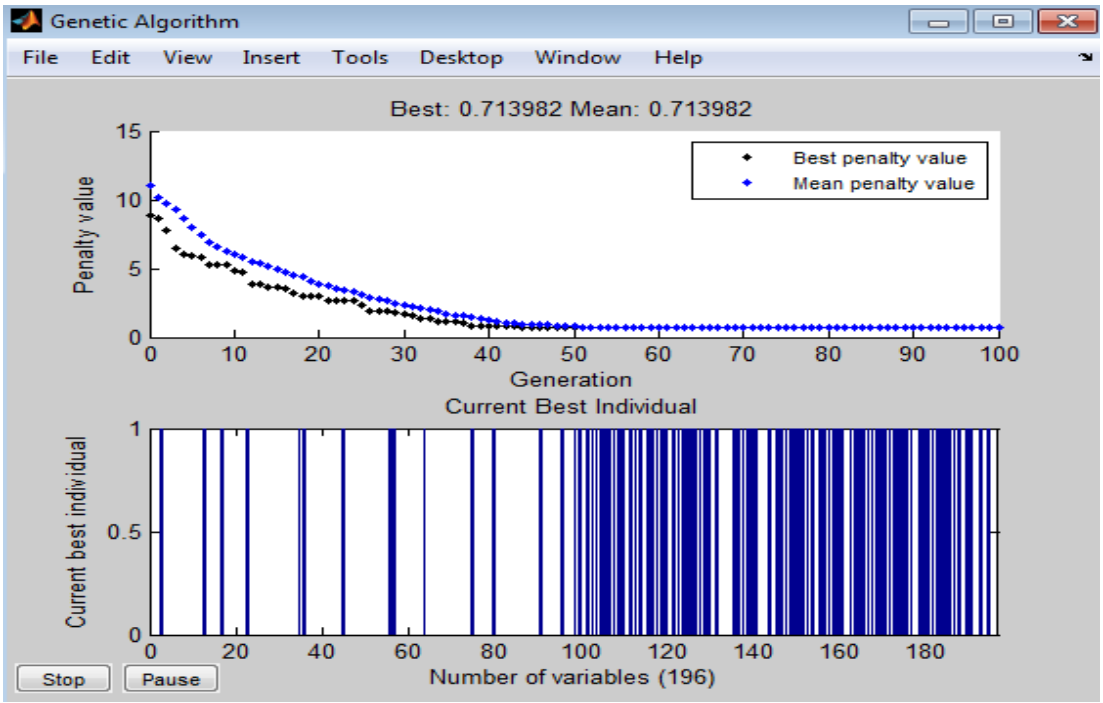

Figure 5-6: Graphical Representation of Best Fitness and Best Individual Results

5.3 Coverage of Aegean Sea

Coefficients of objective function and constraints which are used in experiments for coverage of Aegean Sea are shown in Table 5-10 below.

Table 5-10: Coefficients of Objective Function and Port Constraints

PORT NAME	LOCATION COST OF PORT(UNIT)	COVERAGE DISTANCE OF PORT(NM)	DEPTH OF PORT(M)
AYVALIK	2	12	4
IZMIR	2	12	5
KUSADASI	3	12	4
BODRUM	4	12	3
MARMARIS	2	12	4
FETHIYE	4	12	4

Coefficients of ship constraints which are used in experiments for coverage of Aegean Sea are shown in Table 5-11.

Table 5-11: Coefficients of Ship Constraints

CLASS OF SHIP	ID OF SHIP	SPEED OF SHIP(KTS)	SAFETY DRAFT OF SHIP(M)
6	61	17	4
6	62	17	4
6	67	17	4
6	68	17	4
8	80	27	3
8	81	27	3
8	82	27	3
8	83	27	3
1	101	47	3
1	107	47	3
1	109	47	3
3	302	45	3
3	310	45	3
3	311	45	3
3	313	45	3

The problem cannot be solved by MATLAB optimally under reasonable time limit because of variable and constraint number. Best lower bound on objective value, which is 36, is accepted as best solution.

Assignments which are appeared after genetic algorithm solution are shown in Table 5-12.

Table 5-12: Ship and Port Assignments for Coverage of Aegean Sea

Cost=37 Gap= 2.77 % Time:40 sec.	**61**	**62**	**67**	**68**	**80**	**81**	**82**	**83**	**101**	**107**	**109**	**302**	**310**	**311**	**313**
AYVALIK	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
IZMIR	0	1	0	0	0	0	1	0	1	0	0	0	1	0	0
KUSADASI	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
BODRUM	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
MARMARIS	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1
FETHIYE	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

In Figure 5-7, ship and port assignments for coverage of Aegean Sea is shown on the map of Aegean Sea.

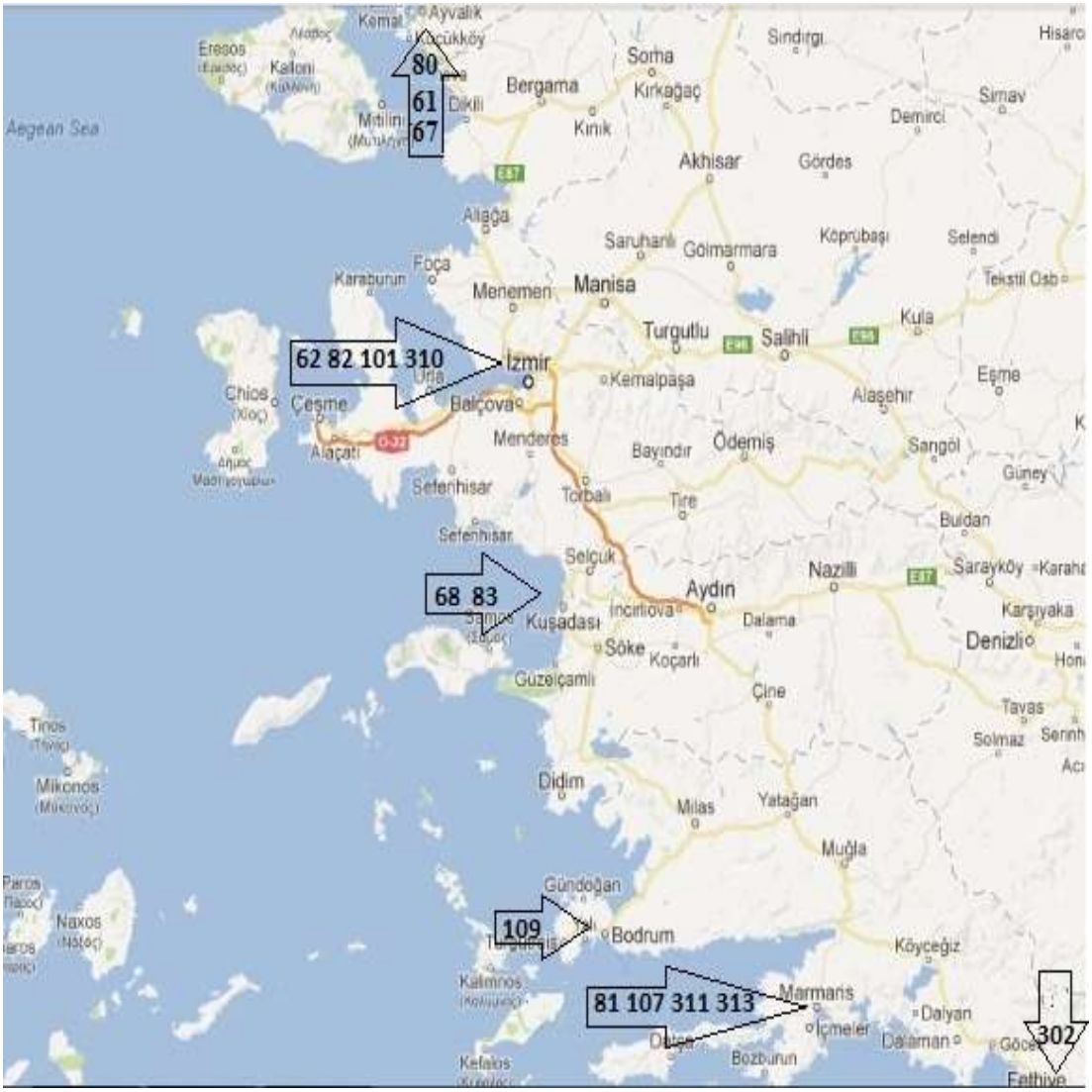

Figure 5-7: Ship and Port Assignments for Coverage of Aegean Sea

In Figure 5-8, graphical representation of best fitness and best individual results of genetic algorithm solution for coverage of Aegean Sea is shown.

In the first graphic, minimization of fitness function values plotted for each generation is shown. Nearly at the half of the all generations, minimization is almost done. In the second graphic, variables which are in the first half of the all variables represents the ships assigned to a port. The lines are 1s and the others are 0s. The second half of the all variables means nothing because they are auxiliary variables.

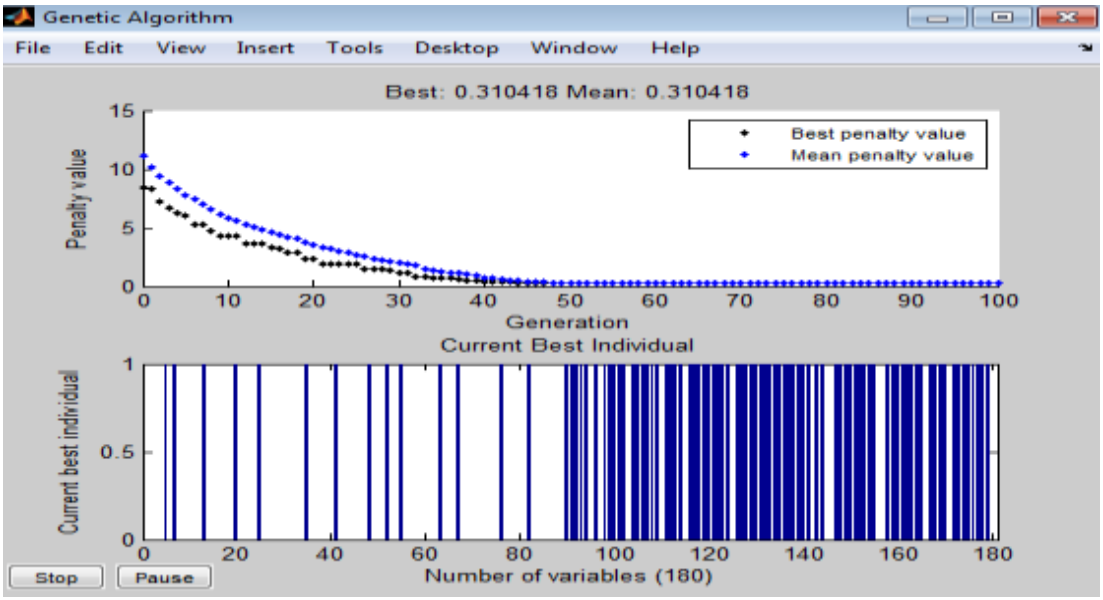

Figure 5-8: Graphical Representation of Best Fitness and Best Individual Results

5.4 Coverage of Mediterranean Sea

Coefficients of objective function and constraints which are used in experiments for coverage of Mediterranean Sea are shown in Table 5-13.

Table 5-13: Coefficients of Objective Function and Port Constraints

PORT NAME	LOCATION COST OF PORT(UNIT)	COVERAGE DISTANCE OF PORT(NM)	DEPTH OF PORT(M)
FINIKE	3	12	3
ANTALYA	2	12	5
ALANYA	4	12	3
BOZYAZI	3	12	3
MERSIN	2	12	5
ISKENDERUN	2	12	3
CEVLIK	3	12	4

Coefficients of ship constraints which are used in experiments for coverage of Mediterranean Sea are shown in Table 5-14 below.

Table 5-14: Coefficients of Ship Constraints

CLASS OF SHIP	ID OF SHIP	SPEED OF SHIP(KTS)	SAFETY DRAFT OF SHIP(M)
6	70	17	4
7	73	30	4
8	89	27	3
8	92	27	3
1	102	47	3
1	103	47	3
1	104	47	3
1	105	47	3
1	108	47	3
3	304	45	3
3	305	45	3
3	306	45	3
3	309	45	3

The problem cannot be solved by MATLAB optimally under reasonable time limit because of variable and constraint number. Best lower bound on objective value, which is 31, is accepted as best solution.

Assignments which are appeared after genetic algorithm solution are shown in Table 5-15.

Table 5-15: Ship and Port Assignments for Coverage of Mediterranean Sea

Cost = 31 Gap= 0 % Time:37 sec.	**70**	**73**	**89**	**92**	**102**	**103**	**104**	**105**	**108**	**304**	**305**	**306**	**309**
FINIKE	0	0	0	0	0	0	0	0	0	1	0	0	0
ANTALYA	0	0	0	0	0	0	0	1	1	0	0	0	0
ALANYA	0	0	0	1	0	0	0	0	0	0	0	0	0
BOZYAZI	0	0	0	0	0	0	0	0	0	0	0	1	0
MERSIN	1	0	1	0	0	0	1	0	0	0	1	0	0
ISKENDERUN	0	1	0	0	0	1	0	0	0	0	0	0	1
CEVLIK	0	0	0	0	1	0	0	0	0	0	0	0	0

In Figure 5-9, ship and port assignments for coverage of Mediterranean Sea is shown on the map of Mediterranean Sea.

Figure 5-9: Ship and Port Assignments for Coverage of Mediterranean Sea

In Figure 5-10, graphical representation of best fitness and best individual results of genetic algorithm solution for coverage of Mediterranean Sea is shown.

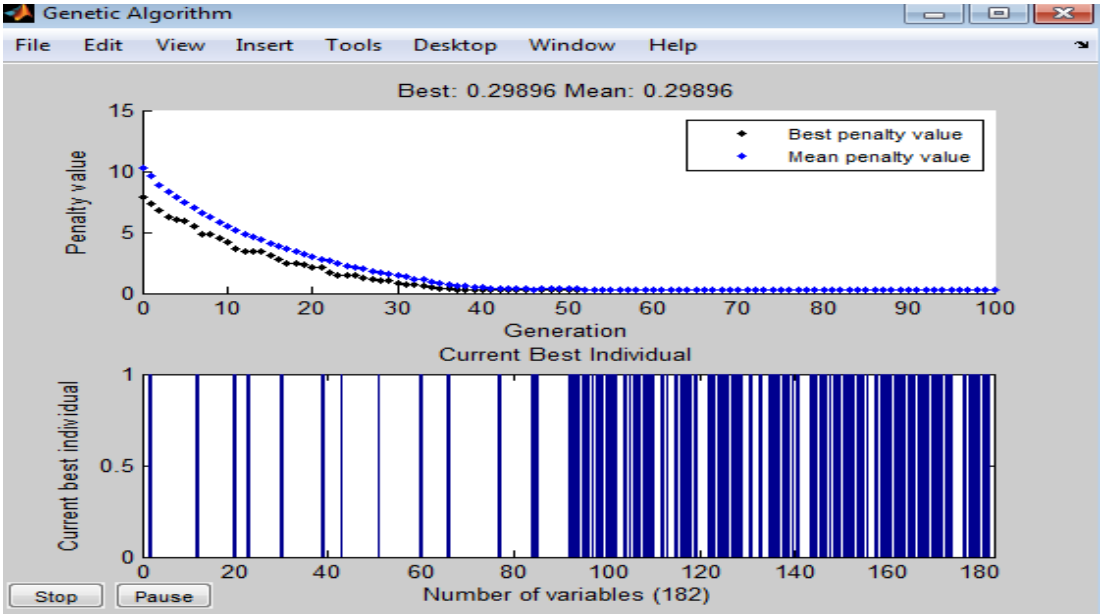

Figure 5-10: Graphical Representation of Best Fitness and Best Individual Results

In the first graphic, minimization of fitness function values plotted for each generation is shown. Nearly at the half of the all generations, minimization is almost done. In the second graphic, variables which are in the first half of the all variables represents the ships assigned to a port. The lines are 1s and the others are 0s. The second half of the all variables means nothing because they are auxiliary variables.

5.6 Comparison of Optimal and GA Solutions

It is clearly shown that genetic algorithm results of coverage of Turkish seas are quite close to best solutions.

The optimality gap ((GA-Op)/Op) between best solution and genetic algorithm solution for Black Sea is 0 %., for Marmara Sea is 3,03 %, for Aegean Sea is 2,77 % and for Mediterranean Sea is 0 %.

In addition to those experiments, some experiments with random values are also done for different sizes of problems. Optimal solution experiments for ship number 9, port number 5 and above cannot be solved by MATLAB under reasonable time limit because of variable and constraint number. Best lower bound on objective value is accepted as best solution for these problems. In Table 5-16, results and comparison of optimal and genetic algorithm solutions are shown.

By examining the table, it is obvious that GA solution is much faster than optimal solution. GA solution is the same as optimal solution in some problem sizes. For others, GA solutions are reasonably close to optimal solutions.

Table 5-16: Comparison of Optimal and GA Solutions

Number of Ships	Number of Ports	Optimal/Best Solution	Op. Sol. Time(sec)	GA Sol.	GA Sol. Time(sec)	% Gap (GA-Op)/Op
5	3	23	18	23	12	0
6	3	24	19	25	12	0
7	3	26	20	27	13	3.84
8	3	27	22	28	14	3.70
6	4	27	812	28	15	3.70
7	4	28	961	28	16	0
8	4	32	3023	32	18	0
9	4	33	4122	34	19	3.03
7	5	29	2221	30	17	3.44
8	5	33	3142	34	18	3.03
9	5	35*	7200	36	21	2.85
10	5	37*	7200	37	23	0
15	10	57*	7200	59	101	3.50

* Best solution after 2 hours of run.

6 Conclusion and Future Study

An overview on the thesis, the motivation for doing this study and the progress of the thesis are explained in chapter 1.

Literature review of covering models, set covering problem (SCP), maximal covering location problem (MCLP) and genetic algorithm (GA) is given in chapter 2.

Progress of maximal covering location problem and relations between the problem worked on in this study and original problem are handled in chapter 3. By explaining the missions and the features of interceptor baywatch vessels, it is expressed how they used in the mathematical model. And, the mathematical model of the problem is defined at the end.

Solution approaches for the problem worked on in this study and how optimal and genetic algorithm solutions are applied on the problem are described in chapter 4.

Experimental study which is done for the problem worked on in this thesis is explained in chapter 5. Solution methodologies expressed in chapter 4 are applied to the problem and experiments for coverage of Turkish seas are done. First parameters are explained and then coverage of all Turkish seas is shown. At the end, comparison of optimal and genetic algorithm solutions is expressed.

In this study, a maximal covering location problem has been considered. It is aimed to solve a real problem which is to develop a new model in order to locate interceptor baywatch vessels for maximal covering of any sea area in this study. It is shown that while genetic algorithm solution is better than the optimal solution in terms of runtime, there are negligible faults compared to the optimal solutions. The heuristic technique produces results average 1.93 % close to the optimal value. Maximum optimality gap is 3.84 %.

The model can be developed and it can be used in solving large-sized defense problems as a future study.

Although genetic algorithm shows really great performance in solving MCLP, another future study can be using different heuristics on this problem.

Appendix A

Glossary of Abbreviations

SCP	Set Covering Problem
MCLP	Maximal Covering Location Problem
LP	Linear Programming
IP	Integer Programming
BIP	Binary Integer Programming
GA	Genetic Algorithm
KTS	Knots
NM	Nautical Miles
M	Meters
SEC	Seconds

Appendix B

Arguments and Options of MATLAB Binary Integer Programming Tool

In this appendix, arguments and options of MATLAB Binary Integer Programming Tool is explained (MATLAB, 2013).

Input Arguments:

f	Vector containing the coefficients of the linear objective function.
A	Matrix containing the coefficients of the linear inequality constraints.
b	Vector corresponding to the right-hand side of the linear inequality constraints.
Aeq	Matrix containing the coefficients of the linear equality constraints.
beq	Vector containing the constants of the linear equality constraints.
x0	Initial point for the algorithm.
options	Options structure containing options for the algorithm.

Problem:	f	Linear objective function vector f
	Aineq	Matrix for linear inequality constraints
	bineq	Vector for linear inequality constraints
	Aeq	Matrix for linear equality constraints
	beq	Vector for linear equality constraints
	x0	Initial point for x
	solver	'bintprog'
	options	Options structure created with optimset

Output Arguments:

Exitflag: Integer identifying the reason the algorithm terminated. The following lists the values of exitflag and the corresponding reasons the algorithm terminated.

1	Function converged to a solution x.
0	Number of iterations exceeded options. MaxIter.
-2	The problem is infeasible.
-4	Number of searched nodes exceeded options. MaxNodes.
-5	Search time exceeded options. MaxTime.
-6	Number of iterations the LP-solver performed at a node to solve the LP-relaxation problem exceeded options. MaxRLP.

Output Structure containing information about the optimization. The fields of the structure are

iterations	Number of iterations taken
nodes	Number of nodes searched
time	Execution time of the algorithm
algorithm	Optimization algorithm used
branchStrategy	Strategy used to select branch variable—see Options
nodeSearchStrategy	Strategy used to select next node in search tree—see Options
message	Exit message

Options:

BranchStrategy Strategy the algorithm uses to select the branch variable in the search tree—see Branching. The choices are

'mininfeas' — Choose the variable with the minimum integer infeasibility (the variable whose value is closest to 0 or 1, but not equal to 0 or 1).

'maxinfeas' — Choose the variable with the maximum integer infeasibility (the variable whose value is closest to 0.5 (default)).

Diagnostics Display diagnostic information about the function. The choices are 'on' or the default, 'off'.

Display	Level of display. 'off' displays no output 'iter' displays output at each iteration 'final' (default) displays just the final output
MaxIter	Maximum number of iterations allowed (a positive integer). The default is 100000*numberOfVariables
MaxNodes	Maximum number of solutions, or nodes, the function searches (a positive integer). The default is 1000*numberOfVariables
MaxRLPIter	Maximum number of iterations the LP-solver performs to solve the LP-relaxation problem at each node (a positive integer). The default is 100*numberOfVariables
MaxTime	Maximum amount of CPU time in seconds the function runs (a positive scalar). The default is 7200.
NodeDisplayInterval	Node display interval (a positive integer). Gives the number of nodes to search between reporting to the iterative display. The default is 20.
NodeSearchStrategy	Strategy the algorithm uses to select the next node to search in the search tree—see Branching. The choices are: 'df' — Depth-first search strategy. At each node in the search tree, if there is a child node one level down in the tree that has not already been explored, the algorithm chooses one such child to search. Otherwise, the algorithm moves to the node one level up in the tree and chooses a child node one level down from that node. 'bn' — Best-node search strategy, which chooses the node with lowest bound on the objective function (the default).

TolFun	Termination tolerance on the function value (a positive scalar). The default is 1.0e-3.
TolXInteger	Tolerance within which the value of a variable is considered to be integral (a positive scalar). The default is 1.0e-8.
TolRLPFun	Termination tolerance on the function value of a linear programming relaxation problem (a positive scalar). The default is 1.0e-6.

- **Bintprog Function**

Binary Integer Programming (bintprog) function uses a linear programming (LP)-based branch-and-bound algorithm for solving binary integer programming problems. The algorithm investigates for an optimal solution to the binary integer programming problem by solving a series of LP-relaxation problems, in which the binary integer requirement on the variables is replaced by the weaker constraint $0 \leq x \leq 1$.

The following sections describe the branch-and-bound method in detail.

- **Branching**

The algorithm creates a search tree by repeatedly adding constraints to the problem that is "branching." At a branching step, the algorithm chooses a variable x_j whose current value is not an integer and adds the constraint $x_j = 0$ to form one branch and the constraint $x_j = 1$ to form the other branch. This process can be represented by a binary tree, in which the nodes represent the added constraints. The following picture illustrates a complete binary tree for a problem that has three

variables, x_1, x_2, and x_3. In general, the order of the variables going down the levels in the tree is not the usual order of their subscripts.

- **Deciding Whether to Branch**

At each node, the algorithm solves an lp-relaxation problem using the constraints at that node and decides whether to branch or to move to another node depending on the outcome. There are three possibilities:

- If the lp-relaxation problem at the current node is infeasible or its optimal value is greater than that of the best integer point, the algorithm removes the node from the tree, after which it does not search any branches below that node. The algorithm then moves to a new node according to the method you specify in nodesearchstrategy option.

- If the algorithm finds a new feasible integer point with lower objective value than that of the best integer point, it updates the current best integer point and moves to the next node.

- If the lp-relaxation problem is optimal but not integer and the optimal objective value of the lp-relaxation problem is less than the best integer point.

The algorithm branches according to the method you specify in the branchstrategy option.

- **Bounds**

The solution to the lp-relaxation problem provides a lower bound for the

binary integer programming problem. If the solution to the lp-relaxation problem is already a binary integer vector, it provides an upper bound for the binary integer programming problem.

As the search tree grows more nodes, the algorithm updates the lower and upper bounds on the objective function, using the bounds obtained in the bounding step. The bound on the objective value serves as the threshold to cut off unnecessary branches.

Appendix C

Arguments and Options of MATLAB Genetic Algorithm Tool

In this appendix, arguments and options of MATLAB Genetic Algorithm Tool is expressed (MATLAB, 2013).

- **Population**

Population options specify options for the population of the genetic algorithm. Population type specifies the type of the input to the fitness function. Types and their restrictions:

- Double vector — Required when there are integer constraints.
- Bit string — For Creation function and Mutation function, use uniform or custom. For crossover function, use scattered, single point, two point, or custom. You cannot use a hybrid function or nonlinear constraint function.
- Custom — For crossover function and mutation function, use custom. For creation function, either use custom, or provide an initial population. You cannot use a hybrid function or nonlinear constraint function.

Population size specifies how many individuals there are in each generation. If you set population size to be a vector of length greater than 1, the algorithm creates multiple subpopulations. Each entry of the vector specifies the size of a subpopulation.

Creation function specifies the function that creates the initial population (ignored with integer constraints):

- Use constraint dependent default chooses:
    - Uniform if there are no constraints
    - Feasible population otherwise
- Uniform creates a random initial population with a uniform distribution.
- Feasible population creates a random initial population that satisfies the bounds and linear constraints.
- Custom enables you to provide your own creation function, which must generate data of the type that you specify in Population type.

Initial population enables you to specify an initial population for the genetic algorithm. If you do not specify an initial population, the algorithm creates one using the Creation function. You can specify fewer than Population size individuals; if you do, the Creation function creates the rest.

Initial scores enable you to specify scores for the initial population. If you do not specify Initial scores, the algorithm computes the scores using the fitness function. Ignored when there are integer constraints.

Initial range specifies lower and upper bounds for the entries of the vectors in the

initial population. You can specify Initial range as a matrix with 2 rows and Initial length columns. The first row contains lower bounds for the entries of the vectors in the initial population, while the second row contains upper bounds. If you specify Initial range as a 2-by-1 matrix, the two scalars expand to constant vectors of length Initial length. Each integer-constrained component has an implicit lower bound of -9,999 and an implicit upper bound of 10,001. Other components have implicit lower and upper bounds of 0 and 1 respectively. Any explicit bounds you give override these implicit bounds.

- **Fitness Scaling**

The scaling function converts raw fitness scores returned by the fitness function to values in a range that is suitable for the selection function. Scaling function specifies the function that performs the scaling. You can choose from the following functions:

- Rank scales the raw scores based on the rank of each individual, rather than its score. The rank of an individual is its position in the sorted scores. The rank of the fittest individual is 1, the next fittest is 2, and so on. Rank fitness scaling removes the effect of the spread of the raw scores.

- Proportional makes the expectation proportional to the raw fitness score. This strategy has weaknesses when raw scores are not in a "good" range.

- Top scales the individuals with the highest fitness values equally. If you select this option, you can specify Quantity, the number of fittest individuals that produce offspring. Quantity must be an integer between 1 and Population

Size or a fraction between 0 and 1 specifying a fraction of the population size. Each of these individuals has an equal probability of reproducing. The rest have probability 0 of reproducing.

- Shift linear scales the raw scores so that the expectation of the fittest individual is equal to a constant, which you can specify as Maximum survival rate, multiplied by the average score.

- Custom enables you to write your own scaling function.

- **Selection**

The selection function chooses parents for the next generation based on their scaled values from the fitness scaling function. You can specify the function that performs the selection in the Selection function field. You can choose from the following functions:

- Stochastic uniform lays out a line in which each parent corresponds to a section of the line of length proportional to its expectation. The algorithm moves along the line in steps of equal size, one step for each parent. At each step, the algorithm allocates a parent from the section it lands on. The first step is a uniform random number less than the step size.

- Remainder assigns parents deterministically from the integer part of each individual's scaled value and then uses roulette selection on the remaining fractional part.

- Uniform select parents at random from a uniform distribution using the

expectations and number of parents. This results in an undirected search. Uniform selection is not a useful search strategy, but you can use it to test the genetic algorithm.

- Shift linear scales the raw scores so that the expectation of the fittest individual is equal to a constant, which you can specify as Maximum survival rate, multiplied by the average score.

- Roulette simulates a roulette wheel with the area of each segment proportional to its expectation. The algorithm then uses a random number to select one of the sections with a probability equal to its area.

- Tournament selects each parent by choosing individuals at random, the number of which you can specify by Tournament size, and then choosing the best individual out of that set to be a parent.

- Custom enables you to write your own selection function.

- **Reproduction**

Reproduction options determine how the genetic algorithm creates children at each new generation.

Elite count specifies the number of individuals that are guaranteed to survive to the next generation. Set Elite count to be a positive integer less than or equal to Population size.

Crossover fraction specifies the fraction of the next generation that crossover produces. Mutation produces the remaining individuals in the next generation. Set

Crossover fraction to be a fraction between 0 and 1, either by entering the fraction in the text box, or by moving the slider.

- **Mutation**

Mutation functions make small random changes in the individuals in the population, which provide genetic diversity and enable the genetic algorithm to search a broader space.

Specify the function that performs the mutation in the Mutation function field. You can choose from the following functions:

- Use constraint dependent default chooses:
    - Gaussian if there are no constraints
    - Adaptive feasible otherwise
- Gaussian adds a random number to each vector entry of an individual. This random number is taken from a Gaussian distribution centered on zero. The standard deviation of this distribution can be controlled with two parameters. The Scale parameter determines the standard deviation at the first generation. The Shrink parameter controls how standard deviation shrinks as generations go by. If the Shrink parameter is 0, the standard deviation is constant. If the Shrink parameter is 1, the standard deviation shrinks to 0 linearly as the last generation is reached.
- Uniform is a two-step process. First, the algorithm selects a fraction of the vector entries of an individual for mutation, where each entry has the same probability as the mutation rate of being mutated. In the second step, the

algorithm replaces each selected entry by a random number selected uniformly from the range for that entry.

- Adaptive feasible randomly generates directions that are adaptive with respect to the last successful or unsuccessful generation. A step length is chosen along each direction so that linear constraints and bounds are satisfied.

- Custom enables you to write your own mutation function that satisfies any constraints specified.

- **Crossover**

Crossover combines two individuals, or parents, to form a new individual, or child, for the next generation. Specify the function that performs the crossover in the Crossover function field. You can choose from the following functions:

- Scattered creates a random binary vector. It then selects the genes where the vector is a 1 from the first parent, and the genes where the vector is a 0 from the second parent, and combines the genes to form the child. For example:

  p1 = [a b c d e f g h]

  p2 = [1 2 3 4 5 6 7 8]

  Random crossover vector = [1 1 0 0 1 0 0 0]

  Child = [a b 3 4 e 6 7 8]

- Single point chooses a random integer *n* between 1 and Number of variables, and selects the vector entries numbered less than or equal to *n* from the first

parent, selects genes numbered greater than *n* from the second parent, and concatenates these entries to form the child. For example:

p1 = [a b c d e f g h]

p2 = [1 2 3 4 5 6 7 8]

Random crossover point = 3

Child = [a b c 4 5 6 7 8]

- Two point selects two random integers *m* and *n* between 1 and Number of variables. The algorithm selects genes numbered less than or equal to m from the first parent, selects genes numbered from *m*+1 to *n* from the second parent, and selects genes numbered greater than *n* from the first parent. The algorithm then concatenates these genes to form a single gene. For example:

  p1 = [a b c d e f g h]

  p2 = [1 2 3 4 5 6 7 8]

  Random crossover points = 3,6

  Child = [a b c 4 5 6 g h]

- Intermediate creates children by a random weighted average of the parents. Intermediate crossover is controlled by a single parameter Ratio:

  child1 = parent1+rand*Ratio*(parent2 - parent1)

If Ratio is in the range [0,1], the children produced are within the hypercube defined by the parents locations at opposite vertices. If Ratio is in a larger range, say 1.1, children can be generated outside the hypercube. Ratio can be a scalar or a vector of length Number of variables. If Ratio is a scalar, all the children lie on the

line between the parents. If Ratio is a vector, children can be any point within the hypercube.

- Heuristic creates children that randomly lie on the line containing the two parents, a small distance away from the parent with the better fitness value, in the direction away from the parent with the worse fitness value.

- Arithmetic creates children that are a random arithmetic mean of two parents, uniformly on the line between the parents.

- Custom enables you to write your own crossover function that satisfies any constraints specified.

- **Migration**

Migration is the movement of individuals between subpopulations, which the algorithm creates if you set Population size to be a vector of length greater than 1. Every so often, the best individuals from one subpopulation replace the worst individuals in another subpopulation. You can control how migration occurs by the following three parameters.

Direction specifies the direction in which migration can take place:

- If you set Direction to Forward, migration takes place toward the last subpopulation. That is the *n*th subpopulation migrates into the (*n*+1)th subpopulation.

- If you set Direction to Both, the *n*th subpopulation migrates into both the (*n*–1)th and the (*n*+1)th subpopulation.

Migration wraps at the ends of the subpopulations. That is, the last subpopulation migrates into the first, and the first may migrate into the last. To prevent wrapping, specify a subpopulation of size 0.

Fraction controls how many individuals move between subpopulations. Fraction is the fraction of the smaller of the two subpopulations that moves. If individuals migrate from a subpopulation of 50 individuals into a population of 100 individuals and Fraction is 0.1, 5 individuals (0.1 * 50) migrate. Individuals that migrate from one subpopulation to another are copied; they are not removed from the source subpopulation.

Interval controls how many generations pass between migrations. If you set Interval to 20, for example, migration between subpopulations takes place every 20 generations.

- **Constraint Parameters**

Constraint parameters refer to the nonlinear constraint solver. Initial penalty specifies an initial value to be used by the algorithm. Initial penalty must be greater than or equal to 1.

Penalty factor increases the penalty parameter when the problem is not solved to required accuracy and constraints are not satisfied. Penalty factor must be greater than 1.

- **Hybrid function**

Hybrid function enables you to specify another minimization function that runs after the genetic algorithm terminates. Not available with integer constraints.

The choices available are:

- None
- fminsearch (unconstrained only)
- patternsearch (constrained or unconstrained)
- fminunc (unconstrained only)
- fmincon (constrained only)

Specify nondefault options for the hybrid function by creating an options structure at the command line with optimset (psoptimset for patternsearch).

- **Stopping Criteria**

Stopping criteria determines what causes the algorithm to terminate. Generations specifies the maximum number of iterations the genetic algorithm performs. Time limit specifies the maximum time in seconds the genetic algorithm runs before stopping.

Fitness limit — If the best fitness value is less than or equal to the value of Fitness limit, the algorithm stops.

Stall generations — If the weighted average change in the fitness function value over Stall generations is less than Function tolerance, the algorithm stops.

Stall time limit — If there is no improvement in the best fitness value for an interval of time in seconds specified by Stall time limit, the algorithm stops.

Function tolerance — If the cumulative change in the fitness function value over Stall generations is less than Function tolerance, the algorithm stops.

Nonlinear constraint tolerance specifies the termination tolerance for the maximum nonlinear constraint violation.

- **Plot Functions**

Plot functions enable you to plot various aspects of the genetic algorithm as it is executing. Each one draws in a separate axis on the display window. Use the Stop button on the window to interrupt a running process.

- Plot interval specifies the number of generations between successive updates of the plot.

- Best fitness plots the best function value in each generation versus iteration number.

- Best individual plots the vector entries of the individual with the best fitness function value in each generation.

- Distance plots the average distance between individuals at each generation.

- Expectation plots the expected number of children versus the raw scores at each generation.

- Genealogy plots the genealogy of individuals. Lines from one generation to the next are color-coded as follows:

    - Red lines indicate mutation children.
    - Blue lines indicate crossover children.
    - Black lines indicate elite individuals.

- Range plots the minimum, maximum, and mean fitness function values in each generation.
- Score diversity plots a histogram of the scores at each generation.
- Scores plots the scores of the individuals at each generation.
- Selection plots a histogram of the parents. This shows you which parents are contributing to each generation.
- Stopping plots stopping criteria levels.
- Max constraint plots the maximum nonlinear constraint violation.
- Custom Enter a function handle of the form @plotfun, where plotfun.m is a function file.

- **Output function**

Output functions are functions that the genetic algorithm calls at each iteration. To use an output function, enter a function handle of the form @outputfun in custom function, where outputfun.m is a function file. You can use multiple output functions by passing a cell array of function handles. Disable output functions by unchecking the custom function check box.

- **Display To Command Window**

Level of display specifies the amount of information displayed in the MATLAB Command Window when you run the algorithm. Choose from the following:

- Off — Display no output.
- Iterative — Display information at each iteration of the algorithm.
- Diagnose — Display information at each iteration. In addition, the diagnostic lists some problem information and the options that are changed from the defaults.
- Final — Display only the reason for stopping at the end of the run.

- **User Function Evaluation**

Evaluate fitness and constraint functions specifies how functions you supply are evaluated:

- In serial means the fitness and constraint functions are evaluated separately at each member of a population
- Vectorized means the fitness and constraint functions for an entire population are each evaluated in one function call
- In parallel means the fitness and constraint evaluations are done on a group of processors.

There are some restrictions on the types of problems that ga can solve when you include integer constraints:

-No linear equality constraints.

-No nonlinear equality constraints.

-Only doubleVector population type.

-No custom creation function (CreationFcn option), crossover function (CrossoverFcn option), mutation function (MutationFcn option), or initial scores (InitialScores option). If you supply any of these, ga overrides their settings.

-GA uses only the binary tournament selection function (SelectionFcn option), and overrides any other setting.

-No hybrid function. GA overrides any setting of the HybridFcn option.

-GA ignores the ParetoFraction, DistanceMeasureFcn, InitialPenalty, and PenaltyFactor options.

-The listed restrictions are mainly natural, not arbitrary. For example: There are no hybrid functions that support integer constraints. So, ga does not use hybrid functions when there are integer constraints.

-To obtain integer variables, GA uses special creation, crossover, and mutation functions.

-You cannot use equality constraints and integer constraints in the same problem. You can try to work around this restriction by including two inequality constraints for each linear equality constraint.

-The genetic algorithm attempts to minimize a penalty function, not the fitness function. The penalty function includes a term for infeasibility. This penalty function is combined with binary tournament selection to select individuals for subsequent generations. The penalty function value of a member of a population is:

- If the member is feasible, the penalty function is the fitness function.
- If the member is infeasible, the penalty function is the maximum fitness function among feasible members of the population, plus a sum of the constraint violations of the (infeasible) point.

References

Altay, Ayça. 2007. Genetik Algoritma ve Bir Uygulama. *Master of Science Thesis*. Institute Of Science. Istanbul Technical University.

Beasley, J.E. and Chu, P.C.. 1996. "A Genetic Algorithm for the Set Covering Problem". *European Journal of Operational Research,*1996, p.392-404.

Berman, Oded and Krass, Dmitry. 2002. "The Generalized Maximal Covering Location Problem". *Computers & Operations Research 29.* p. 563–581.

Carlıoğlu, Serkan. 2005. Ege Bölgesinde Deniz Kuvvetleri Komutanlığına Ait Sahil Gözetleme Radarlarının Yeniden Yerleştirilmesi. *Master of Science Thesis*. Institute Of Science. Gazi University.

Church, Richard and Re Velle, Charles. 1974. "The Maximal Covering Location Problem". *Papers of the Regional Science Association,* Vol. 32, p. 101-118.

Curtin, Kevin, Hayslett-Mc.Call and Karen, Qiu, Fang. 2007. "Determining Optimal Police Patrol Areas with Maximal Covering and Backup Covering Location Models". *Netw Spat Econ ,* p.125–145

Emel, Gül Gökay and Taşkın, Çağatay. 2002. "Genetik Algoritmalar ve Uygulama Alanları". *Uludağ University Journal of the Faculty of Economics and Administrative Sciences,* vol.XXI, no.1, p.129-152.

Farahani, Reza, Asgari, Nasrin, Heidari, Nooshin, Hosseininia, Mahtab and Goh, Mark. 2012. "Covering Problems in Facility Location: A Review". Computers & Industrial Engineering 62. p. 368–407.

Galvao, Roberto Dieguez and Re Velle, Charles. 1996. "A Lagrangean Heuristic for the Maximal Covering Location Problem". *European Journal of Operational Research 88*. p. 114-123.

Gülsün, Bahadır, Tuzkaya, Gülfem and Duman, Cem. 2009. "Facility Layout Design with Genetic Algorithms and an Application". Journal of Dogus University, vol.10, no.1, p.73-87.

Haupt, Randy L. and Haupt, Sue Ellen. 2004. "Practical Genetic Algorithms".2nd ed. *John Wiley & Sons, Inc., Hoboken, New Jersey.*

İşçi, Öznur and Korukoğlu, Serdar. 2003. "Genetik Algoritmalar ve Uygulama Alanları". *Celal Bayar University Journal of Management and Economy,* vol.10, no.2, p.191-208.

Jabalamelia, Mohammad Saeed, Tabrizia, Behzad Bankian and Javadia, Mohammad Moshref. 2011. "A Simulated Annealing Method to Solve a Generalized Maximal Covering Location Problem". *International Journal of Industrial Engineering Computations 2*. p.439-448.

Karaman, Mesut. 2008. A Genetic Algorithm For The Multi-Level Maximal Covering Ambulance Location Problem. *Master of Science Thesis*. The Graduate School Of Natural And Applied Sciences, Middle East Technical University.

Karasakal, Orhan and Karasakal, Esra K. 2004. "A Maximal Covering Location Model in the Presence of Partial Coverage". *Computers & Operations Research 31*. p. 1515–1526.

Laifenfeld, Moshe and Trachtenberg, Ari. 2008. "Identifying Codes and Covering Problems". *IEEE Transactıons on Informatıon Theory,* vol. 54, no. 9, p.3929-3950.

Li, Jingpeng and Kwan, Raymond S K. 2002. "A Fuzzy Evolutionary Approach with Taguchi Parameter Setting for the Set Covering Problem". *Institute of Electrical and Electronics Engineers.* p.1203-1208.

Madhavapeddi, Arvind K.S. 2010. Genetic Algorithm Optimized PID Controllers for a Solid Oxide Fuel Cell System. *Master of Science Thesis*. The Faculty of the Graduate School, Tennessee Technological University.

MATLAB. 2013. Accessed January 28. http://www.mathworks.com/help/

Ratick, Samuel, Osleeb, Jeffrey and Hozumi, Dai. 2009. "Application and Extension of the Moore and ReVelle Hierarchical Maximal Covering Model". Socio-Economic Planning Sciences 43. p. 92–101.

ReVelle, Charles, Scholssberg, Michelle and Williams, Justin. 2008. "Solving the Maximal Covering Location Problem with Heuristic Concentration" Computers & Operations Research 35. p. 427 – 435.

Robles, Danny. 2012. Optimal Signal Control with Multipleobjectives in Traffic Mobility and Environmental Impacts. *Master of Science Thesis*. Royal Institute of Technology.

Seargeant, Daniel Bryan. 2012. The Maximal Covering Location Problem:An Application in Reproductive Health Services. *Doctor of Public Health Thesis,* University of California, Los Angeles.

Silva, Ricardo M. de A. and Ramalho, Geber L. 2001. "Ant System fort the Set Covering Problem". *Institute of Electrical and Electronics Engineers.* p. 3129-3133.

Sivanandam, S.N. and Deepa, S.N., 2008. “Introduction to Genetic Algorithms”. *Springer Berlin Heidelberg, New York.*

Töreyen, Özgün. 2007. Hierarchical Maximal Covering Location Problem With Referral In The Presence Of Partial Coverage. *Master of Science Thesis.* The Graduate School Of Natural And Applied Sciences, Middle East Technical University.

Xia, Li, Xie, Ming, Xu, Weida, Shao, Jinyan, Yin, Wenjun and Dong, Jin. 2009. “An Empirical Comparison of Five Efficient Heuristics for Maximal Covering Location Problems”. *Institute of Electrical and Electronics Engineers.* p. 747-753.

Yin, Ping and Mu Lan. 2012. “Modular Capacitated Maximal Covering Location Problem for the Optimal Siting of Emergency Vehicles”. Applied Geography 34. p. 247-254.

Zarandi, M.H. Fazel, Davari, S., and Sisakht, S.A.H. 2011. “The Large Scale Maximal Covering Location Problem”. Scientia Iranica E. 18 (6), p. 1564–1570.