International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 09, September-2016

Optimization of Berth Scheduling Problem using Genetic Algorithm

Soumen Paul
Dept of Information Technology
Haldia Institute of Technology
Haldia, India

Omprakash Chakraborty
Dept of Information Technology
Haldia Institute of Technology
Haldia, India

***Abstract*- An algorithm is presented for the scheduling of ships into berths using the minimization of net waiting time by Genetic Algorithm. The output as waiting time of the ships are derived based on their operation times and availability of berths as in section V. Then the equivalent service order of ships is determined by minimizing the net waiting time based on threshold comparison in Genetic algorithm. The minimization procedure is simple and computer oriented. It is shown that the algorithm has several advantages, e.g. the reduced waiting time enhances the berth utilization thereby increasing service capacity and sailing time. One numerical example is solved to illustrate the efficiency of the algorithm in berth scheduling problem,**

Keywords— Net Waiting Time, Genetic Algorithm, Threshold, Operation Time

I. INTRODUCTION

Every berth service system can be translated into mathematical model. The mathematical procedure of system modeling often leads to comprehensive description of a process in the form of higher time complexity which is not preferred in terms of both port and sailing efficiencies respectively. It is, therefore, useful, and sometimes necessary, to find the possibility of finding scheduling sequences of the same ships but yielding lower time complexity that may be considered to reflect adequately the optimized service of the system under consideration. Some of the reasons for using scheduling sequence models of port servicing systems could be as follows

- To have a better organising of the system,
- To reduce time complexity,
- To enhance berth capacity,
- To increase sailing time.

Various techniques [1]-[15] have been suggested related to berth scheduling by previous authors. In 2009 [1], three models were described for discrete dynamic berth allocation of arrival of ships. An improved Lagrangian relaxation algorithm [2] was developed to solve the problem of dynamically scheduling ships to multiple continuous berth spaces at the raw material docks. It is implemented in an iron and steel complex with the objective of minimizing the total weighted service. The solution of berth allocation [3] and yard assignment problems was presented by building a collaborative berth allocation model with multiple ports under the background of bulk ports. The neighborhood-search based heuristic optimization approach [4] for the berth scheduling problem was presented to determine the berthing time and space for each incoming ship. A knowledge reasoning mechanism [5] was designed along with numerical experiments for both the berth allocation and quay crane assignment. That illustrates the proposed knowledge-based system. Studying the problem of berth allocation with a priority service [6] by presenting a model of priority along with the simulation of the problem was done through the improvement of the availability of the berths. Genetic Algorithm which is based on the Darwinian principle [7] of natural selection had been successfully applied to Berth allocation problem (BAP), which can decide the ships' berthing position and berthing time at a container terminal. The problem of berth allocation was done [8] by minimization of the total waiting time of the vessels, along with the improvement of the availability of the berths through the presentation of priority model. This average ship waiting time at the berthing area of port container terminal was reduced [9, 10] using queuing theory at ship tugging operation. Heuristic procedure based on genetic algorithm [11] was presented of determining a dynamic berth assignment to ships in the public berth system. A heuristic algorithm [12] for solving discrete berth allocation problem (DRAP) had been evaluated for three different berthing policies. Simulation experiment was developed for this evaluation process. Berth allocation models [13, 14] that allow multiple ships to occupy a single berthing position were explored. DRAP [15] was extended to the multi water depth configuration in a public berth system with proposed Genetic Algorithm. The paper is organized in as follows

In section II the objective of paper is elaborated. Problem formulation with GA is represented in section III. Section IV describes all the algorithms in sequence and flow chart of the algorithm along with an example. In section V, the solution of example I is described and interpreted through tabular and graphical presentation. The limitation of the work and future scope are properly narrated in section VI. Conclusion of the research work is mentioned in section VII.

II. OBJECTIVE

The objectives of this work are:

To develop a model formulating the ship scheduling

- Problem, and find an algorithm to solve this problem.
- To achieve the minimum net waiting time of ships for berth allocation.

- Balanced distribution of service time among all the berths with minimization of net waiting time.

III. THE PROBLEM FORMULATION USING GENETIC ALGORITHM

lim: the total number of chromosomes.
ns: the total number of ships.
nb: the total number of berths available.

$a_{ij}=b_i$ implies that the j^{th} ship order in i^{th} chromosome is assigned to b_i ; where b_i corresponds to the berth having the least service time.
j = (1,2,…,ns)
i = (1,2,…,lim)

$bt_i= opt(a_{jk})$ implies that the operation time of the k^{th} ship of the j^{th} chromosome is added to the service time of the i^{th} allocated berth.
i = (1,2,…,nb)
j = (1,2,…,lim)
k= (1,2,…,ns)

The basics of optimization revolve round the following objective function (Z):
Z= Minimize wt,

where $wt = \sum_{i=1}^{ns} bt_i$, for all i = 1, 2… ns,

where bt_i is the waiting time for individual ships, subjected to the following constraints

$Opt(a_i) \leq th$ for all i = 1, 2, …,lim , where $Opt(a_i)$ denotes the operation time of a_i^{th} chromosome from the parent population of size lim.

$\sum_{j=1}^{nb} x_{ij} = 1$ for all i=1, 2,…,ns (implies one berth can serve only one ship at a time)

$a_i>a_i-1$ for all i=1,2,…,ns (implies that the ships in the chromosome 'a' will be served only in the given order of the chromosome).

$Opt(a_i)$, ns, nb>0, for all i=1,2,…,ns(implies that the operation time of ships in the chromosome 'a' , the number of ships 'ns' and the number of berths(nb) all must have a positive value).

IV. ALGORITHM AND FLOWCHART OF THE PROBLEM

Declaration of variables:

nb: To store the total number of berths.
ns: To store the total number of ships
lim: To store the total number of parent chromosomes
tp: Array to hold the operation time of parent Chromosomes
tps: Array to hold the sorted operation time of parent Chromosomes
epoch: To store the total number of iterations.
th: To store the threshold value
sp: Array to store the selected parents
cnt: To hold Counter value
cp: To hold individual parent chromosomes
newp: Array to store the new population.
newpt: Array to store operation times of the new population
newpts: Array to store the sorted operation times of the new population
a: To store the chromosome
op: Array to hold the operation time of the chromosomes.
wt: OUTPUT variable for the function to return the net waiting time of the chromosome.
b: Array to hold the berth numbers available.
bt: Array to hold the service time at individual berths.
s: To hold the individual ship from a given chromosome.
sp: Array to hold the population and also hold the resultant population to return as OUTPUT
ptc: To hold the population size
tm: To hold the population as in sp temporarily.
cp: To assign the crossover point.
ch: Array to hold the population and also hold the resultant population to return as OUTPUT.
n: To hold the ship numbers

Algorithm: BERTH_SCH1

Step 1: Initialize the values of ns and nb by taking inputs from the user.
Step 2: Repeat step 3 for *'ns'* times.
Step 3: Get the operation times of the individual ships from the user as input.
Step 4: Initialize the value of *'lim'* by receiving the value of the number of parent strings to be used as the initial population.
Step 5: Repeat steps 6 to 9 for *'lim'* times
Step 6: Generate random combination of *'ns'* numbers to develop individual chromosomes.
Step 7: Calculate the operation time of the generated chromosome using *'opt1'* function and them it to the *'tp'* array.
Step 8: Repeat step 9 for *'ns'* times.
Step 9: Add the generated chromosome to the parent population.
Step 10: Repeat steps 11 to 23 for *'epoch'* times.
Step 11: Sort the *'tp'* array in ascending order of operation time in *'tps'*.
Step 12: Initialize the values of *'th'* and *'sp'* to 0 and *'cnt'* value to 1.
Step 13: Calculate the threshold value for the parent population.
Step 14: Repeat steps 15 and 16 for *'lim'* times.
Step 15: Get the parent chromosome in *'cp'*.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 09, September-2016

Step 16: Select the chromosome *'cp'* into *'ps'* if their operation time is less than the threshold value.
Step 17: Use *'crossover'* and *'crossmod'* functions to generate valid offsprings of the parents.
Step 18: Include the new offspring into the *'newp'* array of size *'lim'*.
Step 19: Include the best of parents to fill the remaining space of the array.
Step 20: Display the new population.
Step 21: Calculate the operation times for the new population chromosomes in *'newpt'* and display it's sorted from in *'newpts'*.
Step 22: Assign the new population of *'newp'* as the parent of the next genetic optimization epoch.
Step 23: Assign the operation times of the new chromosomes *'newp'* in the *'tp'*.
Step 24: Display the optimized time result.
Step 25: END.

Algorithm: OPT1

Step 1: Initialize all values of *'b'* and *'bt'* to 0.
Step 2: Repeat steps 3 and 4 for *'nb'* times.
Step 3: Allow the ship to access the berths.
Step 4: Update *'bt'* with the *'op'* of the corresponding ships.
Step 5: Repeat step 6 to 8 for (*'ns'*-*'nb'*) times.
Step 6: Extract the individual ship numbers from the chromosome.
Step 7: Assign the berth having the lowest service time to the next ship.
Step 8: Consider the current service time as of *'bt'* as the waiting time incoming ship.
Step 9: Add the individual waiting time in *'bt'* to the net waiting time of *'wt'*.
Step 10: Return *'wt'*.
Step 11: END

Algorithm: CROSSOVER

Step 1: Repeat step 2 for *'ptc'* times
Step 2: Repeat step 3 for *'ns'* times.
Step 3: Assign the chromosomes of *'sp'* into *'tm'*.
Step 4: Select 2 parent chromosomes sequentially.
Step 5: If *'ns'* is even, then
Calculate *'cp'* = *'ns'*/2.
else
Calculate *'cp'*= (*'ns'*-1)/2.
[end if]
Step 6: Interchange the prior and posterior parts of *'cp'* between the parents and vice-verse with *'sp'* and *'tm'*.
Step 7: Return *'sp'*.
Step 8: END.

Algorithm: CROSSMOD

Step 1: Initialize *'n'* with the values of ship numbers.
Step 2: Repeat step 3 for 'ptc' times.
Step 3: Repeat steps 4 for *'ns'* times.
Step 4: Verify the ships numbers of *'ch'* with that of *'n'*.
Step 5: If number matches with any value of *'n'*, then,
Replace the number with 0.
Else
Replace the number with the index of *'n'* having the first non-zero element.
[end if]
Step 6: Repeat Step 1.
Step 7: Return *'ch'*.
Step 8: END.

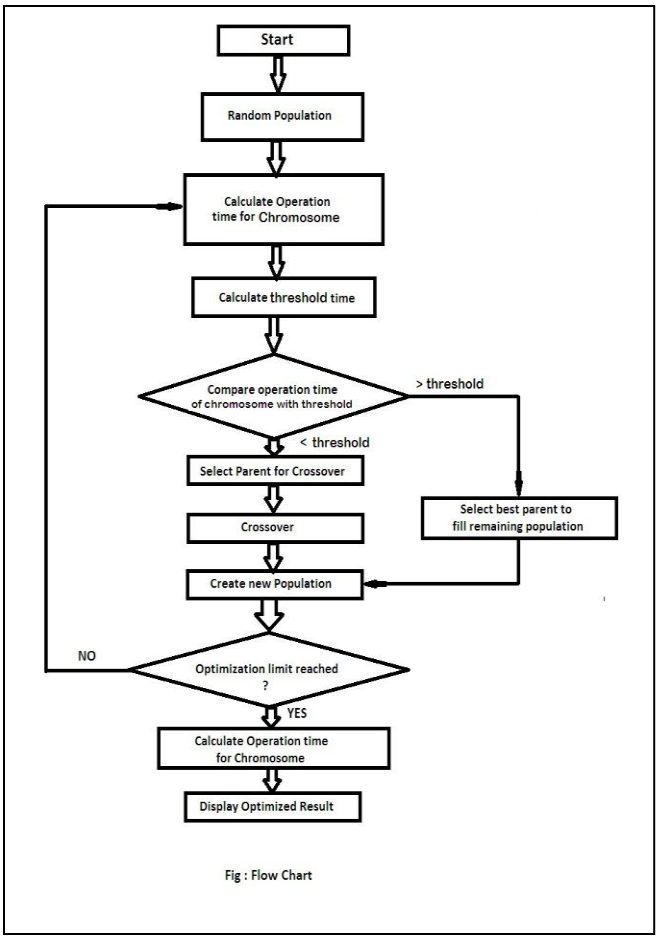


Figure 1: Flow chart of the problem

T

he solution procedure is written in the MATLAB and executed on a personal computer equipped with a Intel® Core™ i5-3230M CPU running at 2.60 GHz speed. The

memory size is 8.00 GB with x 64 bit operation system of Microsoft Windows 8.1.

Example 1:
Let us consider the following problem for berth scheduling problem.Number of ships: 6

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 09, September-2016

Number of berths: 3
Operation time for ship 1: 90 units
Operation time for ship 2: 70 units
Operation time for ship 3: 80 units
Operation time for ship 4: 50 units
Operation time for ship 5: 60 units
Operation time for ship 6: 40 units

I. RESULT AND ANALYSIS

Solution:
The numbers of parent chromosomes are considered as 100.

Table 1: Convergence table of final solution using algorithm BERTH_SCH1

Sl. No.	Epoch	B1	B2	B3	Wt	Sequence/Order of Service
1	1	70	90	80	240	<2,1,3,4,5,6>
2	3	50	90	80	220	<4,1,3,2,5,6>
3	5	70	80	60	210	<2,3,5,1,4,6>
4	7	90	40	60	190	<1,6,5,4,2,3>
5	9	40	50	90	180	<6,4,1,2,3,5>
6	11	80	50	40	170	<3,4,6,1,5,2>
7	13	60	40	50	150	<5,6,4,1,2,3>
8	14	50	60	40	150	<4,5,6,1,2,3>

Thus it is observed that the net waiting time of ships converge at 150 time units keeping balanced service pressure at the 3 berths.

Figure 2: Bar diagram of epoch vs net waiting time

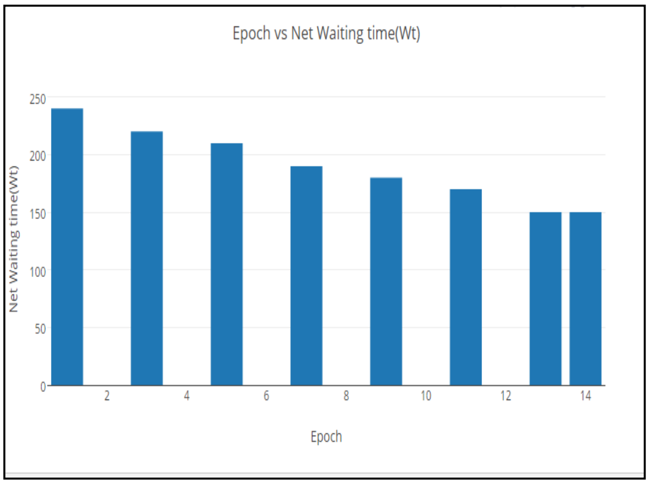
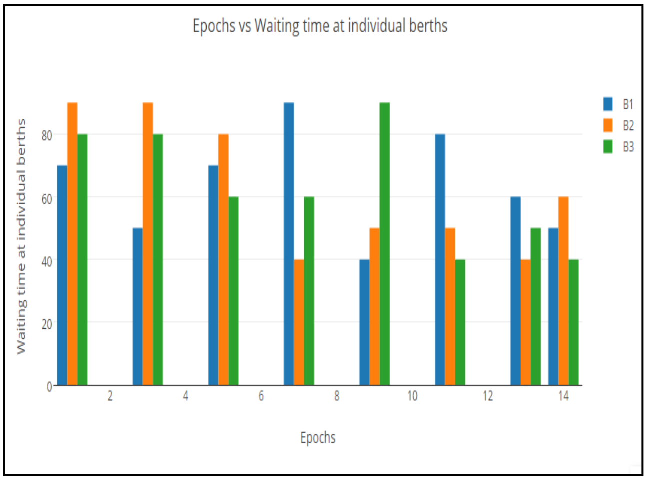


Figure 3: Bar diagram of epoch vs Individual berth waiting time.



Example 2:
No of ship=ns=50
No of berths=nb=30
No of parent chromosome=100

Solution:
The initial normal waiting time is 1818 units.
After optimization the waiting time reduces to 1709 time units.
Epoch =50
Similarly in testing conditions concerning large number of ships(i.e ns=50) in case of a larger port (with 30 berths) the initial normal net waiting time is obtained as 1818 time units which after optimization reduces to 1709 time units.

VI. LIMITATIONS & FUTURE SCOPE

Some factors that are yet to be implemented are:

1. Availability of tug boat
2. Type of ship
3. Traffic congestion
4. Availability of crew
5. Time for Terminal duties
6. Emergencies

With the incorporation of the above features the formulated algorithm can promise much practical and real-world applicable result. It will ease up the concerned authority's need to hire expert experienced personals to monitor scheduling process. However the approach can be modified to learn from real experts by means of different learning algorithms that can work well with genetic algorithmic approach. With the ongoing research on harnessing capabilities of genetic approach in computation there remains a lot of possibility yet to be implemented.

VII. CONCLUSION

The major objective of the computational experiment was to evaluate the Minimum waiting time of the proposed GA designed model in terms of the quality of the solutions and the computing time. This evaluation was achieved by comparing the results of the two approaches: the exact method using FCFS (Manual Computation) and the GA designed model. Computational results also indicate that the optimization approach can solve some moderate size problems by using a method to generate a subset of feasible schedules, which is considered very close to optimal solution. On the other hand, a very large problem will be difficult to be solved due to out of memory, where it recommends resorting to GA for solving them. Meantime, computational results indicate that GA in term of the quality of the solutions is slightly better than work using FCFS method. Experiment results presented for the GA approach indicate that when the number of epoch is large, better solution may obtain. However, some users prefer reasonable solution in less time consumption. Further works can be done by elimination of disadvantages. Finally, the procedure is put to proof in a real scenario in which six ships had to visit three different Berths and serve between them. In the example 1 it is proven that the method brings advantages when compared with FCFS.

REFERENCES

[1] Katja Buhrkal, Sara Zuglian, Stefan Ropke, Jesper Larsenc, Richard Lusby, "Models for the Discrete Berth Allocation Problem: A Computational Comparison", Preprint submitted to Transportation Research Part E August 19, 2009
[2] Lixin Tang, Shaohua Li, and Jiyin Liu, "Dynamically scheduling ships to multiple continuous berth spaces in an iron and steel complex", Intl. Trans. in Op. Res. 16 (2009) pp. 87–107
[3] Nitish Umang, Michel Bierlaire, Ilaria Vacca, ""The berth allocation problem in bulk ports", STRC April, 2011
[4] Yusin Lee, Chuen-Yih Chen, "An optimization heuristic for the berth scheduling problem", European Journal of Operational Research, Vol 196, Issue 2, 16 July 2009, pp 500–508
[5] Sotirios Theofanis, Maria Boile, and Mihalis Golias, "An Optimization Based Genetic Algorithm Heuristic for the Berth Allocation Problem", 2007 IEEE Congress on Evolutionary Computation (CEC 2007), pp. 4439-4445
[6] Tong Shan, "Genetic Algorithm for Dynamic Berth Allocation Problem with Discrete Layout", The 2nd International Conference on Computer Application and System Modeling (2012), pp. 261-264
[7] Charif Mabrouki, Ahmed Faouzi, Ahmed Mousrij, "A Priority Decision Model for Berth Allocation and Scheduling in a Port Container Terminal", Journal of Theoretical and Applied Information Technology 20th August 2013. Vol. 54 No.2, pp. 276-286
[8] A. Shahpanah, S. Shariatmadari, A. Chegeni, A. Gholamkhasi, M. Shahpanah, "Improvement in Queuing Network Model to Reduce Waiting Time at Berthing Area of Port Container Terminal via Discrete Event Simulation", Applied Mechanics and Materials Vol. 621 (2014) pp. 253-258
[9] Wen-Chih Huang, Sheng-Chieh Wu, "The Estimation of the Initial Number of Berths in a Port System Based on Cost Function", Journal of Marine Science and Technology, Vol. 13, No. 1, pp. 35-45 (2005)
[10] Nicolaou, S. N. (1967). ''Journal of the Waterways and Harbors Division, 1967, Vol. 93, Issue 4, Pg. 107-132
[11] Lalla-Ruiz, E.; Melian-Batista, B., and Moreno-Vega, J.M., 2012. Artificial intelligence hybrid heuristic based on tabu search for the dynamic berth allocation problem. Engineering Applications of Artificial Intelligence, 25(6), pp. 1132-1141
[12] S.-W. Lin and C.-J. Ting, "Solving the dynamic berth allocation problems by simulated annealing," Engineering Optimization, vol. 43, no. 3, pp. 308–327, 2014.
[13] G. G. Brown, S. Lawphongpanich, and K. P. Thurman, "Optimizing ship berthing," Naval Research Logistics, vol. 41, no. 1, pp. 1–15, 1994.
[14] G. G. Brown, K. J. Cormican, S. Lawphongpanich, and D. B. Widdis, "Optimizing submarine berthing with a persistence incentive," Naval Research Logistics, vol. 44, no. 4, pp. 301–318, 1997.
[15] E. Nishimura, A. Imai, and S. Papadimitriou, "Berth allocation planning in the public berth system by genetic algorithms,"European Journal of Operational Research, vol. 131, no. 2, pp, 282–292, 2001.

APPENDIX

Source Code

1. File name: BERTH_SCH1.m

```
clc;
clear;
ns = input('Enter no. of ships :');
nb = input('Enter no. of berths :');
for i=1:ns
fprintf('operation time for ship %d : ',i);
op(i) = input('');
end
disp(op);
%op=[60 90 70 50 40 3 80];
%ns=7;
%nb=3;
%a=[4 3 2 5 1 7 6];
%t=opt1(a,op,ns,nb);
%disp(t);
%------------------------------------------------------------
-------------
lim=input('Enter no. of parent chromosomes :');
for x=1:lim
   a=randperm(ns);
tp(x)=opt1(a,op,ns,nb);
disp(a);
for z=1:ns
pch(x,z)=a(z);
end
end
epoch=input('Enter no. of epochs :');
forep=1:epoch
disp(pch);
disp('---------------------------------------------------------
-----------------');
disp(tp);
disp('-------------------------------after sorting----------
-------------------------------');
tps=sort(tp);
disp(tps);
th=0;
sp=0;
th=(min(tps)+max(tps))/2;
disp('Threshold value :');
disp(th);
cnt=1;
for i=1:lim
cp=(pch(i,:));
if opt1(cp,op,ns,nb)<=th;
disp(cp);
ps(cnt,:)=cp;
cnt=cnt+1;
else
continue;
end
end
selp=crossover(ps,(cnt-1),ns);
selp=crossmod(selp,(cnt-1),ns);

newp=selp;
  j=0;
disp('----------------');
disp('----------------');
disp(cnt);
psl=length(ps);
disp(psl);
for i=1:psl
newp(cnt+j,:)=ps(i,:);
    j=j+1;
end
if length(newp)<lim
for i=1:(lim-(length(newp)))
newp(length(newp)+1,:)=pch(i,:);
```

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 5 Issue 09, September-2016

```
end
end
disp('----------------');
disp(newp);
disp('----------------');
forni=1:lim-1
disp(newp(ni,:));
news=newp(ni,:);
newpt(ni)=opt1(news,op,ns,nb);
disp(newpt(ni));
end
newpts=sort(newpt);
disp('----------------');
disp('<><><><><><><><><><><><><><><><><><><><><><><><><><><<
><><><><><>');
pch=newp;
tp=newpt;
end
disp('----------------');
disp(newpts(1));
```

2. File Name:crossmod.m

```
function[ch]=crossmod(ch,ptc,ns)
for i=1:ns
n(i)=i;
end
for i=1:ptc
for j=1:ns
if(n(ch(i,j))~=0)
n(ch(i,j))=0;
else
for z=1:ns
if n(z)~=0
ch(i,j)=n(z);
n(ch(i,j))=0;
break;
end
end
end
end
for i=1:ns
n(i)=i;
end
end
disp(ch);
```

3. File name:crossover.m

```
function[sp]=crossover(sp,ptc,ns)
disp(ptc);
for x=1:ptc
for z=1:ns
tm(x,z)=sp(x,z);
end
end
if(mod(ns,2)==0)
cp=ns/2;
for i=1:2:ptc-1
for j=1:cp
sp(i,j)=tm(i+1,cp+j);
sp(i+1,cp+j)=tm(i,j);
end
end
end
if(mod(ns,2)~=0)
cp=(ns-1)/2;
for i=1:2:ptc-1
or j=1:cp
sp(i,j)=tm(i+1,cp+j);
sp(i+1,cp+j)=tm(i,j);
end
end

end
disp(sp);
```

4. File name:opt1.m

```
function[wt]=opt1(a,op,ns,nb)

for i=1:nb
b(i)=0;
bt(i)=0;
end
for i=1:nb
b(i)=a(i);
bt(i)=op(a(i));
end
for i=(nb+1):(ns-nb)
 s=a(i);
```