Yönetim, Yıl: 18, Sayı: 58, Ekim 2007

A GENETIC ALGORITHM APPROACH TO DETERMINE SHIPPER SIZES FOR STORAGE

Dr. Tunçhan CURA

İstanbul Üniversitesi

İşletme Fakültesi

Sayısal Yöntemler Anabilim Dalı

There has been a lot of research into transportation. But most of this is focused on transportation network. This study investigates how containers are loaded as a part of transportation. For this reason, the shippers in which the products are allocated are very important. In order to minimize the sum of the transportation volume, a genetic algorithm approach has been exploited in this study. Random values example has been made up to generate a great size of example with respect to the real world values. As a result, it has been stated that the algorithm produces significant results.

Key Words: Transportation, genetic algorithms, shipper, storage.

DEPOLAMA IÇIN PAKET BÜYÜKLÜĞÜNÜN BELİRLENMESİNE YÖNELİK BİR GENETİK ALGORİTMA YAKLAŞIMI

Taşımacılık üzerine literatürde çok sayıda araştırma yapılmıştır. Ancak bu araştırmaların pek çoğu taşımacılık ağı odaklıdır. Bu çalışmada ise taşımacılıkta konteynırlara yüklemenin nasıl yapılacağı incelenmektedir. Bunun için ürünlerin yerleştirildiği, yüklemede kullanılacak paketler son derece önemlidir. Toplam taşıma hacmini asgari düzeye indirmek amacıyla genetik algortimalardan yararlanılmıştır. Büyük ölçekli olması için gerçek hayata uygun tesadüfi değerlerden bir örnek oluşturulmuştur. Sonuç olarak algoritmanın anlamlı sonuçlar ürettiği tespit edilmiştir.

Anahtar Sözcükler: Taşımacılık, genetik algoritmalar, sevkiyat paketi, depolama.

INTRODUCTION

As long as the cost of transportation is high, it is not enough to reduce the cost of the production. In addition to this, a very important element of the transportation costs is packaging. The ongoing drive to reduce packaging waste has led to an interest in the use of multi-trip containers or shippers. The usual practice with disposable containers is to use a bespoke carton for each product. However, in a multi-use system it is often necessary to reduce this to a relatively small number in order to maintain a manageable and cost effective logistics system for the containers themselves (Dowsland, 2005, 1). Thus, the right choice of shipper dimensions becomes very important.

In this study, the right set of shippers is chosen by a genetic algorithm approach. The model has been based on the model which is developed by Dowsland, Soubeiga and Burke (2005). Total volume has been minimized depending on annual demands. In this manner, not only packaging waste would be reduced but also the number of containers or the size of containers would be reduced. Shintani, Imai, Nishimura and Papadimitriou (2007) proposed an integrated and genetic algorithm focused comprehensive approach which optimizes the transportation problem by designing the network. This study only deals with the storage of containers neglecting the design of network of transportation. Alicke (2002) investigated the intermodal terminal concept. In his study the terminal was modeled as a multi-stage transshipment problem. In his approach, sequence-dependent duration of empty moves, alternative assignments (of containers to cranes) and a sequence-dependent number of operations were handled. An optimization model based on Constraint Satisfaction was formulated and heuristics for the search procedure, especially value and variable ordering were developed. Although there are many studies for minimization of transportation costs problem, it is clearly seen that research on determining shipper sizes for storage are very rare.

In this paper, a genetic algorithm approach is developed and it is used as optimization tool. Genetic algorithms (GA) were developed initially by Holland and his associates at University of Michigan in the 1960s and 1970s, and the first full, systematic (and mainly theoretical) treatment was contained in Holland's book Adaptation in Natural and Artificial Systems published in 1975 (Reeves, 1995, 152).

1. THE PROBLEM

According to Dowsland, Soubeiga, and Burke (2005) the problem is stated as follows: The number of different products is denoted by n. Each product ($i=1,n$) has four characteristics of length, width, height and weight, denoted by x_i, y_i, z_i, and w_i respectively. Each product also has an expected annual sales volume in terms of number of units sold, denoted by b_i. The objective, for a given integer p, is to find a set of p totes or shippers $j(j=1, p)$ to be used for storage and/or distribution of the products. The dimensions of shipper j is represented by variable X_j, Y_j and Z_j. As explained previously, the problem is to minimize the total volume of shippers required. It is assumed that each individual shipper is filled with a single product (although shippers of a given set of dimensions may obviously be used for several different products). Thus, the optimization model for the problem is formed as:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{m} \beta_{ij} x_{ij}$$

$$s.t \quad \sum_{j=1}^{m} x_{ij} = 1,\ i = 1, n$$

$$x_{ij} \le y_j, \quad i = 1, n,\ j = 1, m$$

$$\sum_{j=1}^{m} y_j = p, \quad p \in \mathrm{N}$$

$$x_{ij}, y_i \in (0,1)$$

where $x_{ij} = 1$ if product i is allocated to shipper j, $x_{ij} = 0$ otherwise, $y_j = 1$ if shipper j is used, $y_j=0$ otherwise, and $\beta_{ij} = \dfrac{V_j b_i}{f_{ij}}$ *where* $V_j = X_j \times Y_j \times Z_j$ and f_{ij} is the number of units of product i that will fit into shipper j. It will be calculated as follows:

$$f_{ij} = \min\left\{\left\lfloor \frac{W_{\max}}{w_i} \right\rfloor, \left(\left\lfloor \frac{Z_j}{z_i} \right\rfloor \times \left\lfloor \frac{X_j}{x_i} \right\rfloor \times \left\lfloor \frac{Y_j}{y_i} \right\rfloor\right)\right\}$$

where W_{max} is the maximum weight per shipper. In order to search solutions for great size of problems, β_{ij}'s are set randomly with respect to the real world values in this study.

2. DETERMINIG THE POPULATION SIZE and THE NUMBER OF GENES

In this problem, it is tried to pick out the most appropriate p shippers from m shippers. This means the selection is made within $\binom{m}{p}$ alternatives. Thus, each

chromosome has p genes and each shipper is denoted by a gene. In order to give the chance of being selected to each of m shippers, all of them must take place in the initial population. Thus, the population size is calculated as $\left\lceil \frac{m}{p} \right\rceil$ in this study. When $m = 10^5$ and p=10 the population will consist of 10^4 chromosomes.

A subject that has attracted little attention is how to select the initial population. Usually, this is simply chosen by generating random chromosomes (Reeves, 1995, 166). While setting up the initial population, each gene must have randomly unique values between 1 and m. In this manner, each shipper has a chance to take place in crossover operation. As the number of generations progressively increases, the genes which cause better fitness values will possibly repeat in current population.

In order to avoid the possibility of having all the chromosomes same in any population of further generations - although genes can repeat - the uniqueness of each chromosome must be ensured. For example if c_1={1, 3, 5, 2} and c_2={4, 1, 6, 2} then c_1's first gene is the same as c_2's second gene and c_1's last gene is the same as c_2's last gene. However, these two chromosomes are not the same.

As another example, if it is assumed that m = 16, and p = 3 then the population size will be 6 and the initial population may be as follows:

c_1 = {2, 16, 3} c_2 = {7, 12, 9} c_3 = {11, 15, 1}
c_4 = {8, 13, 6} c_5 = {5, 10, 14} c_6 = {4, 2, 1}

3. SELECTION MECHANISM

In Holland's original GA, parents are selected by means of stochastic procedure from the population, and a complete new population of offsprings is generated and then these offsprings replaced with their parents. In another version, he suggested that each offspring should replace a randomly chosen member of the current population as it was generated (Reeves, 1995, 166). One simple method is the roulette wheel approach for selection mechanism. Each individual is assigned to a slot whose probability is proportional to the fitness of that individual, and the wheel is spun each time a parent is needed (Masters, 1993, 144).

In this study a similar approach is applied: Since the problem is a minimization, the lesser an individual has fitness value, the more it should have possibility to be selected. In order to enable this, each individual's selection possibility (s_k) is calculated as follows:

$$s_k = \frac{\sum_{p=1}^{\alpha} f_p}{f_k} \Big/ \sum_{p=1}^{\alpha} \frac{\sum_{l=1}^{\alpha} f_l}{f_p} \qquad f_k > 0 \quad and \quad k = 1, \alpha$$

where α denotes the population size and equals to $\left\lceil \frac{m}{p} \right\rceil$, f_p denotes the fitness value of individual p in the current population. For example if α = 3 and f_i= {10, 15, 20} then s_k will be calculated as follows.

$$\sum_{p=1}^{\alpha} f_p = 10 + 15 + 20 = 45$$

$$\sum_{p=1}^{\alpha} \frac{\sum_{l=1}^{\alpha} f_l}{f_p} = \frac{45}{10} + \frac{45}{15} + \frac{45}{20} = 9.75$$

$$s_1 = \frac{45}{10} \Big/ 9.75 = 0.46 \quad s_2 = 0.31 \quad s_3 = 0.23$$

4. FITNESS CALCULATION

It has been explained that each chromosome consists of shippers where products will be allocated to. Thus each gene is a shipper and product i will be allocated to the gene (shipper) within the current chromosome which has the minimum β_{ij} value. After allocating all the products to the most appropriate genes, the sum of β_{ij}'s will be fitness value for the current chromosome. The algorithm for fitness calculation of chromosome c_1 is as follows:

```
fitness = 0
For each i, where i ≤ n, and i ∈ N
{   j = c1,1
    best = βij
    for each k, where k ≤ p, and k > 1, and k ∈ N
    {   j = c1,k
best = min{best, βij}
        }
    fitness = fitness + best
}
```

5. CROSSOVER OPERATION

After reproduction, each two of the parent strings in the mating pool are picked randomly and each pair of strings undergoes crossover with a probability. Crossover requires two individual chromosomes to exchange their genetic compositions. Thus, the offspring inherits some genes from parents via such operations (Jiao, 2007, 1787).

Each individual chromosome is a set of shippers where the products will be allocated to. Thus, having a chromosome which has repeated genes is needless moreover it is false. As mentioned previously, it has been guaranteed that each chromosome has unique gene in initial population. Since the crossover operator produces new offsprings, in other words, new population, these offsprings must also be guaranteed that they have unique genes individually.

Also if one of the parents is fitter than the both offsprings, this parent must be put into new population. Unfit offsprings might cause next generations go worse.

In this study, a single-point crossover operator is used. The cutting point is selected randomly between 2 and p. The crossover algorithm is as follows.

μ = a random number between 2 and p
$offspring_1 = parent_1$
$offspring_2 = parent_2$
for each i, where $i < \mu$, and $i > 0$, and $i \in N$
{ $offspring_{3,i} = parent_{1,i}$
$offspring_{4,i} = parent_{2,i}$
}
for each i, where $i \geq \mu$, and $i \leq p$, and $i \in N$
{ $offspring_{3,i}$ = *geneFromparent*($offspring_3$, $parent_2$, i)
$offspring_{4,i}$ = *geneFromparent*($offspring_4$, $parent_1$, i)
}
calculate fitness values for each *offspring*
select the *offspring* which has the minimum fitness value and then put it into the new population

Where μ denotes cutting point. To guarantee that an offspring inherits unique genes from a parent, a function *geneFromparent*(*offspring, parent, x*) is developed and it is as follows.

continue = 1
$k = x$
while *continue* is 1
{ $l = 1$
for each i, where $i < x$, and $i > 0$, and $i \in N$
{ if $parent_{k \bmod p}$ is equal to $offspring_i$ then $l = 0$ }
if l is 1 then the result is $parent_{k \bmod p}$ and *continue* = 0
$k = k + 1$
}

For a small size of example it is assumed that $p = 5$, and $parent_1 = \{3, 5, 11, 2, 4\}$, and $parent_2 = \{1, 2, 10, 9, 6\}$, and $\mu = 3$ then

$offspring_1 = \{\underline{3}, \underline{5}, \underline{11}, \underline{2}, \underline{4}\}$ $offspring_3 = \{\underline{3}, \underline{5}, 10, 9, 6\}$

$offspring_2 = \{1, 2, 10, 9, 6\}$ $offspring_4 = \{1, 2, \underline{11}, \underline{4}, \underline{3}\}$

where the underlined genes are from $parent_1$ and the others are from $parent_2$

6. MUTATION

Mutation is an operator which is applied to each offspring individually after crossover. It randomly picks a gene within each string with a small probability (referred to as mutation rate) and alters the corresponding attribute level at random. This process enables a small amount of random search, and thus ensures that the GA search does not quickly converge at a local optimum. But it should not occur very often, otherwise the GA becomes a pure random search method (Jiao, 2007, 1788). Thus, in this study it only occurs with the possibility of ",2" and if the current offspring, produced by crossover, is not unique. But what if the mutation possibility does not come true nevertheless the offspring is unique? In this case the algorithm must keep trying crossover with new randomly selected parents for producing current offspring till the offspring becomes unique or the mutation possibility comes true.

When mutation occurs, a number of genes are randomly picked within current string. This number is called mutation point count in this study. It is calculated as $\left\lfloor \frac{m}{600} \right\rfloor$. The divisor 600 is obtained empirically. This number will increase as the number of shippers increases and that becomes very beneficial in terms of computation time. Indeed, this number can not be higher than p.

$$\theta = \min\left\{\max\left\{\left\lfloor \frac{m}{600} \right\rfloor, 1\right\}, p\right\}$$

Produce vector λ which has randomly selected θ mutation points
$v = 0$
for each i, where $i > 0$, and $i \leq \theta$, and $i \in N$
{ $l = 1$
while l is 1
{ v = random number between 1 and m
$l = 0$
$j = \lambda_i$

for each k, where $k > 0$, and $k \leq p$, and $k \neq j$, and $k \in N$

{ if $offspring_k$ is equal to v then $l = 1$ }

$offspring_j = v$

}

}

If m is 1300 and $p = 5$ and current offspring = {2, 500, 109, 86, 850}, and $\theta = 2$, $\lambda = \{1, 3\}$ then the offspring can be {128, 500, 1208, 86, 850} where the underlined genes are changed after the mutation.

7. THE MAIN ALGORITHM

$$\alpha = \left\lceil \frac{m}{p} \right\rceil$$

Produce initial population c

Calculate fitness values of each chromosome in c

Set selection probabilities of each chromosome in c

Set *bestchromosome* equal to chromosome most appropriate in c

Set *bestfitness* and *previousfitness* equal to minimum fitness value in c

unchangeditertions = 0

generation = 1

While *generation* ≤ 1000

{ $i = 1$

While $i \leq \alpha$

{ Select $parent_1$ and $parent_2$ from c with respect to selection probabilities

Produce new offspring by crossover operator

Set $temporarypopulation_i$ equal to the new offspring

If $temporarypopulation_i$ is not unique in *temporarypopulation* and randomly

selected number < 0.2 then mutate $temporarypopulation_i$

If $temporarypopulation_i$ is unique in *temporarypopulation* then $i = i + 1$

}

Copy the chromosomes from *temporarypopulation* to c

Set selection probabilities of each chromosome in c

If *previousfitness* = minimum fitness value in c then

unchangeditertions = *unchangeditertions* + *1* else *unchangeditertions* = 0

If *unchangeditertions* = 100 then *generation* = 1001

Set *previousfitness* is equal to minimum fitness value in c

If *bestfitness* > minimum fitness value in c then *bestfitness* = minimum fitness value

in c and set *bestchromosome* equal to chromosome most appropriate in c

}

If the minimum fitness value does not change for 100 generations one after the other, the algorithm will stop running for further searches. In this study maximum generation count is set to 1000.

8. DISCUSSION

The algorithm developed in this study is tested with a randomly created problem. The results of the GA are compared with that of Lingo4 software. Because of the software's constraints and variable capacity, in the test problem m, n, and p are set to 1332, 6, and 6 respectively. As explained above, β_{ij} is produced randomly with respect to the real world values. This problem's solution space consists of $\binom{1332}{6} = 7{,}67 \times 10^{15}$ alternatives. Thus, even if any given test computer can experiment 100,000,000 alternatives per second, the total search time of the solution space will be approximately 2.4 years. The lingo model to solve the problem is as follows:

```
MODEL:
SETS:
        SHIPPERS / 1..1332 / : Y;
        PRODUCTS / 1..6 / ;
        ALLOC(PRODUCTS, SHIPPERS)  : X, β;
    ENDSETS

    DATA:
β = 2.7458204×10^7, 1.8495946×10^7, 1.3202413×10^7, 1.4766266×10^7, 1.8574984×10^7, 2.076052×10^7,
        ……
    ENDDATA

        MIN = @SUM(ALLOC: X * β);
        @FOR( PRODUCTS( I): @SUM( SHIPPERS( J): X(I, J)) = 1);
        @FOR(ALLOC(I, J): X(I, J) <= Y(J));
        @SUM(SHIPPERS: Y) = 6;
        @FOR( ALLOC: @BIN(X));
        @FOR( SHIPPERS: @BIN(Y));
    END
```

And the results are as follows:

Optimal solution found at step: 348

Objective value: 0.1426174×10^8

Branch count: 0

Variable Value Reduced Cost

```
    ⋮          ⋮          ⋮
Y( 183)    1.000000    0.0000000
Y( 262)    1.000000    0.0000000
Y( 467)    1.000000    0.0000000
Y( 662)    1.000000    0.0000000
Y( 1093)   1.000000    0.0000000
Y( 1196)   1.000000    0.0000000
    ⋮          ⋮          ⋮
X( 1, 183) 1.000000    2500060.
X( 2, 262) 1.000000    2782824.
X( 3, 662) 1.000000    2301684.
X( 4, 1093) 1.000000   2329361.
X( 5, 1196) 1.000000   2257784.
X( 6, 467) 1.000000    2090027.
```

The GA solution for that problem is obtained in 1.84 seconds on a Pentium IV 1.7. Figure 1 shows the minimum fitness values obtained in each generation.

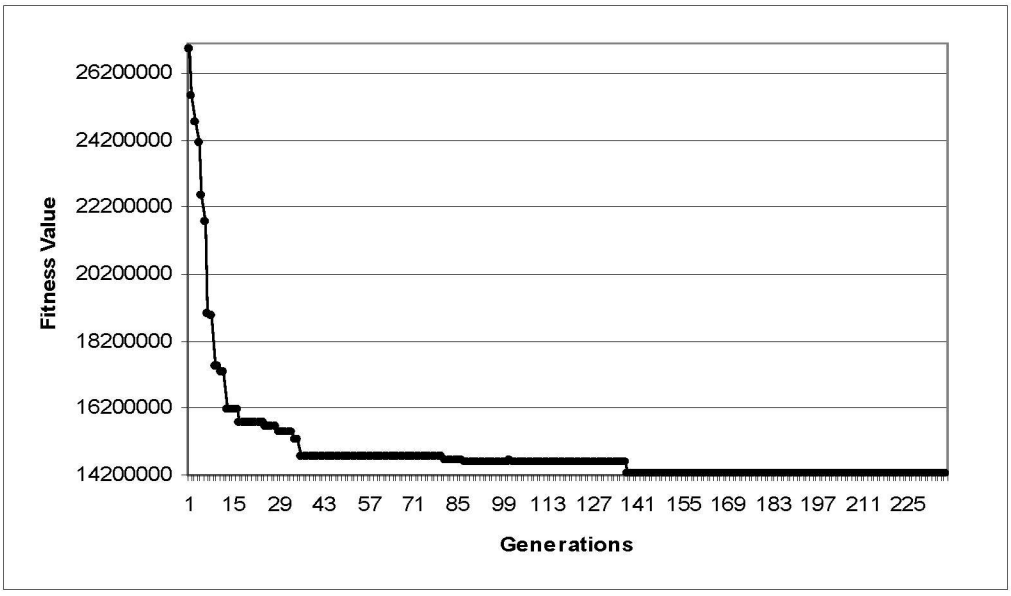

Figure 1. GA generations of the problem

Table 1. GA solution for the problem

PRODUCTS \ SHIPPERS	183	262	467	662	1093	1196
1	X					
2		X				
3				X		
4					X	
5						X
6			X			

Table 1 shows the GA solution for the problem. It is seen that Lingo4 software and GA produce the same solution.

Although the determining the shipper sizes for storage problem is not identical with the 0-1 knapsack problem, they seem similar. The knapsack problem with n items is described by the knapsack of size b and three sets of variables related to the items: decision variables $x_1, x_2, \ldots, x_n$; positive weights $W_1, W_2, \ldots, Wn$; and profits $P_1, P_2, \ldots, P_n$; where, for each $1 \leq i \leq n$, x_i is either 0 or 1. The W_i and P_i represent the weight and profit, as integers, of the ith item, respectively (Kumar ve Banerjee, 2006, 109).

The knapsack problem formally can be stated as follows:

$$\max \quad \sum_{i=1}^{n} P_i x_i$$

$$s.t. \quad \sum_{i=1}^{n} W_i x_i \leq b$$

$$x_i \in \{0,1\}$$

Having been adapted into the knapsack problem, this studies' algorithm is tested with the cases of weing1, weing2, weing3, weing4, weing5, weing6, weing7, weing8, pb4 and hp1 as well. The solutions of the adapted algorithm are the same as the known solutions of which the objective function values are 141278, 130883, 95677, 119337, 98796, 130623, 1095445, 624319, 95168 and 3418, respectively. The test data are obtained from *http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/mknap2.txt*.

9. CONCLUSION

The GA approach in this study produces significant results as tested above. The overall size of the test problem was $7{,}67 \times 10^{15}$ because of Lingo4's capacity. But the performance must be examined with a much greater problem in size. For this reason, a problem where $m = 10^5$, $p = 10$, and $n = 50$ is simulated. Figure 2 shows the progress of algorithm. It lasts 33,67 minutes on the same computer and it is acceptable for this kind of problems. Table 2 shows the solution.

As the memory capacity on subject computer increases, which is 256MB currently, problems which have greater search spaces may be solved. But such size of a search space is assumed to be enough for a real world problem example.

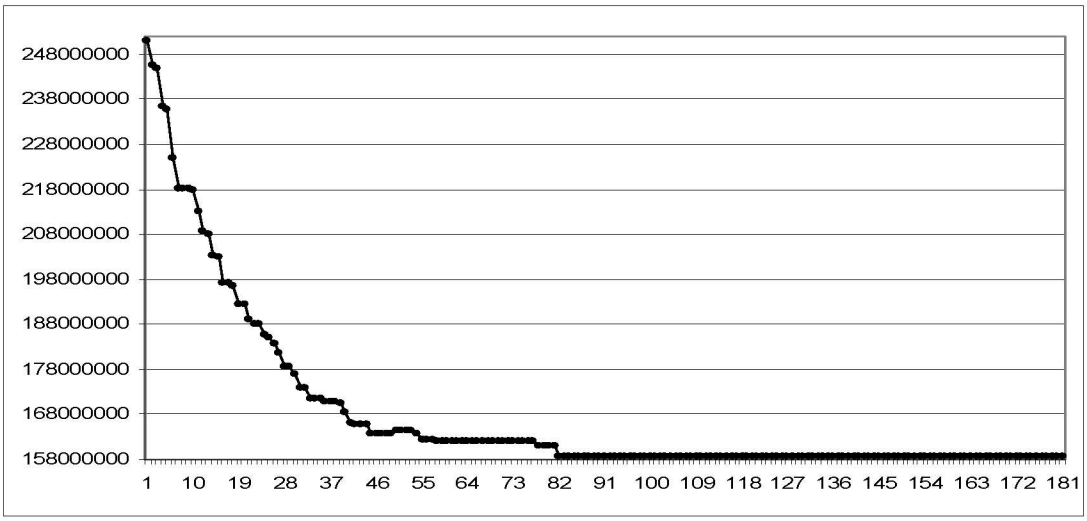

Figure 2. GA generations of great size problem

Table 2. GA solution for the great size problem

PRODUCTS	14859	15985	38578	59897	62274	64541	83363	92485	93353	96372
1								X		
2		X								
3							X			
4									X	
5				X						
6						X				
7								X		
8				X						
9	X									
10										X
11						X				
12									X	
13	X									
14							X			
15			X							
16					X					
17	X									
18		X								
19		X								
20	X									
21		X								
22					X					
23						X				
24		X								
25						X				

PRODUCTS	14859	15985	38578	59897	62274	64541	83363	92485	93353	96372
26			X							
27										X
28				X						
29					X					
30	X									
31							X			
32										X
33				X						
34					X					
35		X								
36									X	
37		X								
38			X							
39							X			
40			X							
41									X	
42		X								
43						X				
44					X					
45				X						
46						X				
47										X
48			X							
49								X		
50										X

REFERENCES

Alicke, K., 2002, *"Modeling and optimization of the intermodal terminal Mega Hub"*, **OR Spectrum,** 24, 1-17

Dowsland, K. A., Soubeiga, E., Burke, E., 2005, *"A simulated annealing based hyperheuristic for determining sipper sizes for storage and transportation"*, **European Journal of Operational Research,** (Baskıda)

http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/mknap2.txt

Jiao, J., R., Zhang, Y.,Wang, Yi, 2007, *"A heuristic genetic algorithm for product portfolio planning"*, **Computers & Operations Research,** 34, 1777-1799

Kumar R.,Banerjee N, 2006, *"Analysis of a Multiobjective Evolutionary Algorithm on the 0–1 knapsack problem"*, **Theoretical Computer Science**, 358, 104 - 120

Masters, T., 1993, **Practical Neural Network Recipes in C++,** Academic Press, USA

Reeves, C., 1995, **Modern Heuristic Techniques for Combinatorial Problems**, McGraw-Hill Book C., London

Shintani, K., Imai, A., Nishimura, E., Papadimitriou, S., 2007, *"The container shipping network design problem with empty container repositioning"*, **Transportation Research Part E,** 43, 39-59