"Vasile Alecsandri" University of Bacău
Faculty of Sciences
Scientific Studies and Research
Series Mathematics and Informatics
Vol. 23 (2013), No. 1, 213 – 224

DOUBLE HYBRIDIZED EMBRYONIC GENETIC ALGORITHM FOR OPTIMAL DELIVERY ROUTING

ADRIAN VÎLCU

Abstract. This paper presents an efficient embryonic hybrid genetic algorithm for finding a minimum cost Hamiltonian circuit for a variable weight transportation problem. Prior work addresses the design of different heuristic methods, pure genetic algorithms or some hybridizations of genetic algorithm with greedy methods. The proposed algorithm includes a combination between a genetic algorithm, a branch and bound method and a dynamic-programming-inspired method. The basic components of the embryonic genetic algorithm are defined, the ways of using dynamic programming as a hypermutation operator and the realization of the fusion between the branch and bound approach and the basic genetic algorithm are described. The experimental results are compared to those given by other existing methods. The originality of this paper results from a combination of exact techniques - branch and bound method and dynamic programming and genetic algorithm. Except a better solving method for the delivery problem, this new solving technique has design implications on the creation of new methods for solving other difficult practical problems.

1. INTRODUCTION

This paper aims to optimize a special type of delivery. In this problem, a vehicle leaving a warehouse, charged with the total demand, reaches once each client satisfying its demand and returns at the departure point.

Keywords and phrases: delivery problem, embryonic genetic algorithm, branch and bound, dynamic programming.
(2010) Mathematics Subject Classification: 68W99.

The transportation cost between two successive locations is a linear function on the transported charge. The problem consists in the finding order to supply the clients that minimizes the transportation cost. Since this problem generalizes traveling salesmen problem (TSP) it is called generalized traveling salesman problem (GTSP).

Approaches to solving this problem can be grouped into two classes: exact methods class - dynamic programming algorithm, branch-and-bound algorithms, and algorithms that use techniques of linear programming type and heuristics and evolutionary algorithms class – constructive heuristics, iterative improvement techniques and randomized improvement algorithms: genetic algorithm, simulated annealing, Tabu search, ant colony optimization, and the cross entropy method.

The novelty of this paper is the fusion between the branch and bound approach and a genetic algorithm on which dynamic programming is grafted as a hypermutation operator.

Section 2 formulates the problem, section 3 describes B&B technique, section 4 presents dynamic programming method, section 5 describes the main components of the embryonic algorithm and the results of the experiments and some conclusive remarks are present in the last section.

2. STATEMENT OF PROBLEM

Let $G=(V,A)$ be a complete and directed graph, where $V=\{0,1,...,n\}$ is the set of vertices, node 0 represents the warehouse, and nodes $1,...,n$ represent the customers, A is the arcs set. The integer $q(i)$ is the demand of the consumer $i, i=1,2,...,n$. The total demand is $Q=q(1)+...+q(n)$. The unitary cost of the transportation charge from node i to node j is $a(i,j)$. The positive integer $q(0)$ represents the empty vehicle mass. The cost of moving charge r from i to j is given by $g(r,i,j)=(r+q(0))*a(i,j)$.

Let us denote by $c=[0,i_1,...,i_n,0]$ a Hamiltonian circuit in G starting from "0", passing through $i_1,...,i_n \in V\setminus\{0\}$ and ending at "0". The cost of this circuit is

$$\text{cost}(c)=g(Q,0,i_1)+\sum_{k=1}^{n-1} g\left(Q-\sum_{h=1}^{k} q(i_h), i_k, i_{k+1}\right)+q(0,i_n,0)$$

If H is the set of all Hamiltonian circuits in G, then GTSP refers to finding $c^* \in H$ so that $\text{cost}(c^*)\le \text{cost}(c),(\forall)\, c\in H$.

Because the TSP is NP-complete [1],[2], the GTSP complexity is also NP-complete.

3. OUTLINE OF B&B TECHNIQUE

B&B method organizes the space solution as a tree $T(N,E)$ where N is the set of nodes and E .is the edges set. The root of the tree corresponds to the whole set of solutions H . A nonterminal node z on the level k of this tree corresponds to a simple path $(z_1,...,z_k)$ in G. The set $S(z) \subset H$ contains all solutions that start with $(z_1,...,z_k)$. A terminal node in T corresponds to complete solution of the problem. The exploration of T is done using an estimate function that associates a lower edge of bound costs of routes in $S(z)$, to each node. Search strategy is the least cost, which means the nonterminal node, which has the smallest value of the estimation function, is branched. The method stops when the minimum is reached for a terminal nod.

In Figure 1 is represented a part of the T tree.

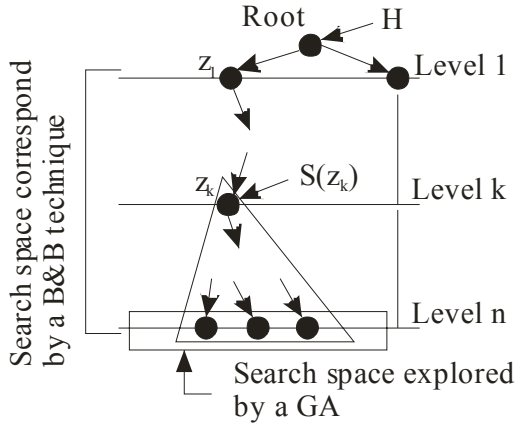


Figure 1 The search space for B&B and Genetic algorithms

The branch techniques are: first, the level one contains the node labels that can be touch from Root node and the second, the descendants of a node $z \in \{k\}, k = 1...n-2$ are $y_1,...y_k$ where $y_j \in V \setminus \{ \text{nodes from } [root,...,z] \text{ path}\}$.

As a common feature, both B&B and GA explore the T tree. The difference is that B&B examines the whole T tree while GA is recruiting its population only from terminal nodes of T .

For the z_k node, the B&B estimation function has two terms, $e(z_k) = e_c(z_k) + e_p(z_k)$. The value $e_c(x_k)$ is the cost of the path $(\text{Root},...,z_k)$ and $e_p(z_k)$ is a lower bound of the costs that correspond to all extension of z_k to complete solutions.

4. OUTLINE OF DYNAMIC PROGRAMMING INSPIRED METHOD

This method was obtained by considering that constructing the solution of the problem appears as a Markov decision process with finite horizon approach [3][4].

We make the following notations:

- i stage index, i = 0,1,…,n+1;
- Xi set of vertices that candidate for position i in the final solution, i = 1,…,n;
- Ai(x) set of predecessors of x∈ Xi belonging to Xi-1, i = 1,…,n;
- Bi(x) subset of Ai(x) containing those vertices that use x as an optimal decision in a previous stage;
- Di(x) set of vertices considered as feasible decisions to reach x, Di(x) = Ai(x)-Bi(x) ;
- Gi(x) minimal cost of accessing x∈ Xi from a vertex belonging to Di(x);
- di*(x) vertex of Di(x) that leads to Gi(x);
- ri(x) remaining charge after arriving at x∈ Xi from di*(x)
- [0, x1*,…, xn*,0] final solution given by the heuristic.

Algorithm is described below:

1. Initialization:

1.1. Determine the set $X_1=\{i|(0,i) \in U\}$.

1.2. For each $x_1 \in X_1$, compute $G_1(x_1) = g(Q; 0,x_1)$, $r_1(x_1) = Q- q(x_1)$ and take $d_1^*(x1) = 0$.

2. For i=2,3,…,n do:

2.1. For each x∈ X-{0} perform:

2.1.1. Determine the set $A_i(x) = \{j \mid j \in X_{i-1}-\{x\}, (j, x) \in U\}$.

2.1.2. Determine the set $B_i(x)$ containing the vertices $j \in A_i(x)$ for which there exists a path from *x* to *j*, $[x,y_1,\ldots,y_k, j]$ so that $d_{i-1}^*(j)=y_k$, $d_{i-2}^*(y_k)=y_{k-1}$, …, $d_{i-k-1}^*(y_1)=x$ (using *j* a premature circuit's closing is produced).

2.1.3. Determine the set $Di(x) = Ai(x)-Bi(x)$.

2.2. Determine the set $X_i=\{x \mid D_i(x) \neq \varnothing\}$

2.3. For each xi∈ Xi compute: $Gi(xi) = Gi-1(di^*)+g(ri-1(di^*); di^*,xj) = \min\{Gi-1(di) + g(ri-1(di); di,xj) \mid d \in Di(xi)\}$ and take $di^*(xi) = di^*$ (optimal decision to reach xi).

2.4. Determine the sequence [xi, xi-1*,..., x2*], where xi-1*= di*(xi), xi-2*= di-1*(xi-1*), …, x2* = d3*(x3*) and ri(xi) = ri-1(xi-1*) -q(xi).

3. Final selection:

3.1. Take $X_{n+1} = \{0\}$, $D_{n+1}(0) = \{j \mid j \in X_n, (j,0) \in U\}$.

3.2. Compute $G_{n+1}(0) = G_n(d_{n+1}^*) + g(r_n(d_{n+1}^*); d_{n+1}^*,0) = \min\{G_n(d_{n+1}) + g(r_n(d_{n+1}); d_{n+1},0) \mid d_{n+1} \in D_{n+1}(0)\}$ and take $d_{n+1}^*(0) = d_{n+1}^*$.

4. Output solution:

4.1. Compute $x_n^* = d_{n+1}^*(0)$, $x_{n-1}^* = d_n^*(x_n^*),\ldots, x_1^* = d_2^*(x_2^*)$.

4.2. Output the circuit $[0, x_1^*,\ldots, x_n^*, 0]$ and its cost $G_{n+1}(0)$.

5. Stop.

The basic step is illustrated in Figure 2. This heuristic is further used for a delivery problem where the source can be different from the destination. The algorithm can be used in the context where the prefix and/or the suffix of the permutation are fixed and a portion of it, forming a contiguous area, is to be given as at output of this algorithm.

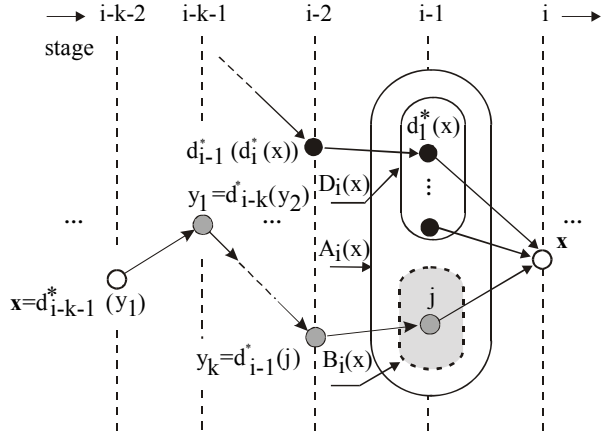


Figure 2. The basic step of heuristic algorithm

V. BASIC COMPONENTS OF THE HYBRID GENETIC ALGORITHM

First, the B&B technique is combined with GA [4], resulting in an embryonic GA. The second hybridization is done with the dynamic-programming-inspired heuristic that acts as a hypermutation operator.

5.1 Solution representation

A chromosome is a sequence $x=(x_1,...,x_k)$ of vertices in G with $1\le k\le n$, where $[0,x_1,...,x_k]$ represents a simple path in G. If $k<n$, x is an embryo, otherwise ($k=n$) x is an adult (complete solution).

5.2 Population management

During all the evolution stages the population size is constant.

The initial population P is composed of 90% embryos $(x_1,...,x_k)$ where k is a random number $k\in[2,n]$. The remaining 10% solutions from initial population are solutions provided by dynamic-programming-inspired heuristic applied on GTSP instances that correspond at random sequences of k nodes from G.

The selection of individuals at the end of each stage is elitist, the $|P|$ best fitness chromosome being selected for the next stage.

The hybrid GA stops if the ratio $\frac{m_k - m_{k-1}}{m_k} < \varepsilon$ where m_k is the average of the fitness at k stage, ε is the tolerance, and the algorithm executes at least $10\% * nr\_iterations$ with complete solutions where $nr\_iteration$ is the number of stages with embryos.

5.3 Fitness function

In fact, the fitness function is the B&B estimation function (Figure 3). For $x = (x_1,...,x_k)$ $e(x) = e_c(x) + e_p(x)$. The term $e_p(x)$ includes a transport from x_k to a node that does not exist in prefix, the moving on the simple path among the location in $R(x) = V \setminus \{x_1...,x_k\}$ and the return from node that are not in prefix to $"0"$: $e_p(x) = e_1(x) + e_2(x) + e_3(x)$.

If the chromosome is adult $(k = n)$ then $cost(x) = e_c(x)$

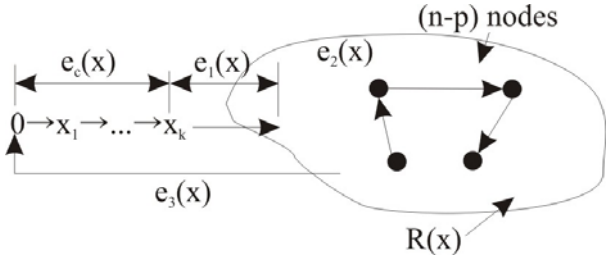


Figure 3. Components of fitness function

The $e_c(x)$ term is given by $e_c(x) = g(Q,0,x_1) + \sum_{k=1}^{p-1} g\left(Q - \sum_{h=1}^{k} q(i_h), i_k, i_{k+1}\right)$. Terms $e_1(x)$ and $e_3(x)$ are defined by $e_1(x) = \min_{\substack{u \in R \\ (x_k,u) \in A}} \{g(Q(x), x_k, u)\}$ where $Q(x) = Q - \sum_{i=1}^{k} q(x_i)$, and $e_3(x) = \min_{\substack{u \in R \\ (u,0) \in A}} \{g(0,u,0)\}$. The term $e_3(x)$ is resulted from the following steps:

Take the first $n-k-1$ smallest unitary cost of the arcs (u,v) with $u,v \in R(x)$, $\alpha_1 \le \alpha_2 \le ... \le \alpha_{n-k-1}$

Sort in descending order the demands of the vertices in $R(x)$, $q(y_1) \ge q(y_2) \ge ... \ge q(y_{n-k})$

Calculate $e_2(x) = \sum_{i=1}^{n-k-1} \alpha_i \left[Q(x) - \sum_{h=1}^{i-1} q(y_h) \right]$

Numerical example. The matrix of unitary cost are:

a	0	1	2	3	4	5	6	7	8	9	10	q(i)
0	0	1	10	1	10	10	1100	1000	110	1000	10	20
1	10	0	10	9	1	511	1100	1000	1000	5000	1	3
2	110	1	0	2000	999	110	5	6	1	19	1	7
3	2	1	1	0	1	176	10	110	4	5	4	10
4	1	10	10	100	0	1	10	1	10	6	8	8
5	3	1	111	1	1	0	10	1	6	1e+10	1	3
6	2	10	10	1	10	7000	0	1	10	5	10	67
7	1	1	3	200	10	4	50	0	10	189	1	3
8	9	10	110	2	2	25	4	90	0	40	1	56
9	1	5	4	2	30	5	48	4	5	0	10	100
10	3	3	10	2	9000	4005	5000	6000	600	70	0	12

Consider $x=(1,10,2,6,9)$, the term $e_c(x)=289*a(0,1)+286*a(1,10)+$ $+274*a(10,2)++267*a(2,6)+200*a(6,9)=5650$.

Now, $e_1(x)=100*min\{a(9,3),a(9,4),a(9,5),a(9,7),a(9,8)\}=200$ and $e_3(x)=20*a(4,0)=20$

For $e_2(x)$, the remaining demands in descending order are $\{56,10,8,3,3\}$ and the first smallest cost of the arcs having both ends in the $R(x)$ are $\{1,1,1,1\}$. It result that

$e_2(x)=(100-56)*1+(100-56-10)*1+(100-56-10-8)*1+(100-56-10-8-3)*1=127$

Finally, $e(x)=5650+200+127+20=5997$

5.4 Mutation operator

For each chromosome from population is generated a random number $nr\in[0,1]$ and if $nr\le p_m$ where p_m is mutation probability, then to this chromosome is applied the mutation operator.

Let $x=(x_1,...x_k)$ be a chromosome on which one can apply mutation operator. It randomly generates two cuts p_1 and p_2, $p_1,p_2\le k$. Let t be a randomly generated integer so that $t<|p_2-p_1|$. Run t circular permutations of the genes in the interval $(min(p_1,p_2),max(p_1,p_2))$. As an example, consider: $x=(3,5,7,2,8,10,9)$, $p_1=6$ and $p_2=2$. The offspring is $x'=(3,8,10,5,7,2,9)$

5.5 Crossover operator

The crossover operator is applied on chromosomes selected from the entire population with the probability p_c. Let two chromosome be $x=(x_1,...,x_k)$, $y=(y_1,...,y_h)$ and t a random number with $t<min(k,h)$ The offspring will be

composed from t length intervals, filled alternately with genes from both parents, without duplicates. For the second offspring, the roles of parents are interchanged.

Numerical example. Consider $x=(3,5,7,2,8,10,9)$, $y=(8,2,10,5,6,9)$ and $t=3$. The offspring are:

$$x_1=(|3,5,7,|8,2,10,|9,6) \text{ and } y_1=(|8,2,10,|3,5,7,|6,9)$$

Since the content of the offspring is the reunion of parents' genes, the crossover operator acts as a growing operator.

5.6 Growth operator

It consists in extending the embryo x with the node $u\in R(x)$ where $\frac{q(u)}{a(x_k,u)}=max\left\{\frac{q(v)}{a(x_k,v)},v\in R(x),(x_k,u)\in A\right\}$.

Numerical example. Consider $x=(3,5,7,2,8,10,9)$. $R(x)=\{1,4,6\}$. The extension node is 6, $\frac{q(6)}{a(9,6)}=max\left\{\frac{3}{5};\frac{8}{30};\frac{67}{48}\right\}=\frac{67}{48}$. Thus, x is replaced with $x'=(3,5,7,2,8,10,9,6)$

5.7 Hypermutation operator

The hypermutation is applied with the probability p_h. Consider a chromosome $x=(x_1,...x_k)$ and two cutting positions r and s, with $r<s\le k$. Consider a subgraph G' generated by $N=\{x_r,...,x_s\}$. The heuristic method applied to G', returns the solution $\{x_r'...,x_s'\}$ which is a permutation of N. Finally, $x_r'...,x_s'$ replace the vertices $x_r,...,x_s$ i.e. the offspring is $x'=[x_1,...,x_{r-1},x_r',...,x_s',x_{s+1},...,x_k]$

Numerical example. Consider $x=(3,|5,7,2,10,|8,9)$ with $\text{cost}(x)=211638$. Assume that $r=2$ and $s=5$. The offspring is $x'=(3,|5,7,10,2,|8,9)$ with $\text{cost}(x')=61290$.

5.8 Performance evaluation

In experiments the benchmarks found at www.misp.tuiasi.ro/it/benchmark_GTSP are used. The experiments were organized in three phases.

In the first, various probability values $\{p_m,p_c,p_h,p_g\}$ were tested in order to find adequate fitness values. For this, the hybrid GA was run 20 times for each combination of probabilities $p_m=\{0.1,0.15,0.2\}$, $p_c=\{0.2,0.35\}$ $p_h=\{0.1,0.15,0.2\}$ $p_g=\{0.2,0.3\}$.Observe that two classes of probabilities provide relatively good solutions with good variances (table 1).

Table 1. Probability classes used into second experimental phase

Classes	p_m	p_c	p_h	p_g	*Average*	*Deviation*	$\frac{Deviation}{Average}$	*Minimum*
I	0.1	0.35	0.2	0.3	83670.23	2233.15	0.026690	80009
II	0.1	0.2	0.1	0.2	84328.27	2382.81	0.028245	78599

The second series of the experiments were done in order to compare the results provided by four algorithms: GA, GA with hypermutation (GAh), GA with B&B (GAb&b) and GA with hypermutation and B&B (GAhb&b). Table 2 shows the average (avg) of the best fitness, the standard deviation (σ) and coefficient of variation σ /avg obtained in 30 runs for these algorithms.

Table 2. Statistical values for algorithms

Algorithm	Average	Deviation	σ /avg	Best fitness
GA	159263.22	12874.08	0.080835	138681
GAh	79514.23	518.33	0.006519	78420
GAb&b	90723.90	2427.85	0.026761	86685
GAhb&b	83670.23	2233.15	0.026690	80009

Best value is obtained for GAh from the heuristic solution injected into initial population on which mutation, crossover and hypermutation operators are applied. This is normal, because besides the fact that the heuristic method gives a very good solution as a global solution, this also has good sequences of genes which are propagated by the operators of mutation, crossover and hypermutation. Except GAh, GAhb&b provides good solutions in terms of best fitness, variance and coefficient of variation.

The third set of experiments studies the behavior of embryonic GA. In this context the influence of hypermutation operator was watched; the results are represented in Figure 4.

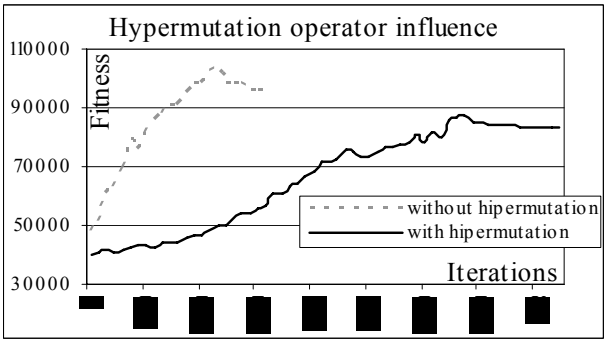


Figure 4. Hypermutation operator influence

The effect of hypermutation operator is a better final solution (80009 vs 86685) and a slow convergence (1180 iterations vs 440 iterations).

In Figure 5 is presented the variation of cost function of the best chromosome to the end of each stage (principal axis OY), and the length of the best chromosome in terms of cost function based on the number of iterations (secondary axis OY') (Figure 5)

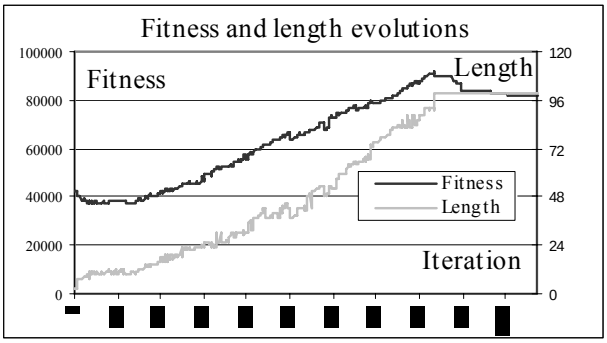
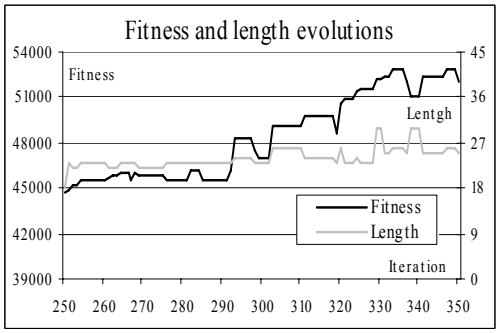
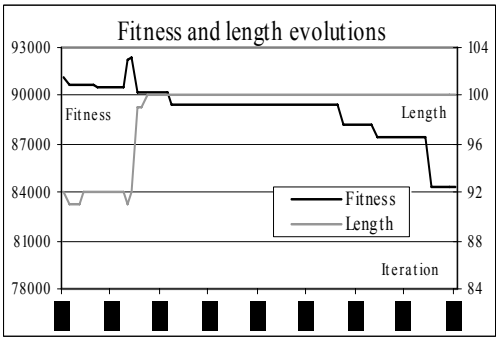


Figure 5. Fitness and length of the best chromosome depending on the number of iterations

The next two charts are zooms on two periods of evolution of the embryonic GA, observing the synchronization between fitness and length of the best chromosome. Notice that the cost function reaches a maximum before the iterations in which all chromosomes are adults (Figure 6 (b)). This phenomenon is explained by the fact that the predictive component of the cost function provides results close to the real cost function (the lengths of chromosomes are approximately equal with the size of the problem) and the crossover operator is able to provide greater length chromosomes with a better fitness.



a



b

Figure 6. Fitness and length of the best chromosome depending on the number of iterations

In Figure 6 (a) is presented the behavior AGhb&b when the hypermutation operator was applied with different probabilities pH=0, 0.05, 0.08, 0.1, 0.15, 0.2

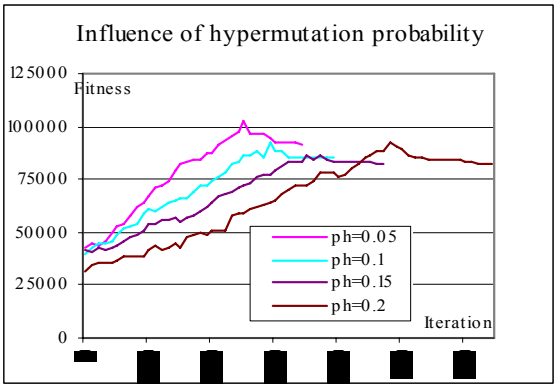


Figure 7. Influence of hypermutation probability

In Figure 7 is observed the beneficial role of the hypermutation operator, which leads to the good quality result in terms of the final solution when it is more present in the evolution algorithm (higher probability). There is still an upper limit of this increase in the value of hypermutation probability; for example, for a class I of probabilities ($p_m = 0.1, p_c = 0.35, p_g = 0.3$), an increase in value over 0.2 will lead to unacceptable calculation times and even to the termination of the program execution in local minimum points - incomplete chromosomes with a very good fitness.

6. CONCLUSIONS AND FURTHER RESEARCH DIRECTIONS

A new and efficient hybrid method resulted from the grafting of two techniques, branch & bound and dynamic programming on a genetic algorithm for solving a variable weight transportation problem was presented. Specific genetic operators including a new type of growing operator acting on partial representations of the solutions are proposed.

The evolving mechanism operates with subsets of solutions and has very good capabilities to locate the better potential zones of the search space early and thus better exploit them.

The efficiency of grafting the B&B method on a genetic algorithm and using dynamic programming method as genetic operator led to an efficient tool due to an appropriate tradeoff between exploration and exploitation.

Experimental investigation has shown that the performances of this new hybrid method are good and are better than those of the basic genetic algorithms.

The impact of these results is larger because these design elements can be used to combine other variants of such algorithms within transportation paradigm as well as for other optimization problems. This type of hybrid GA can also be used in more sophisticated genetic algorithms, like cellular or segregative GA. This is the subject of some further research directions. More accurate fitness evaluation of embryos is also of real interest.

REFERENCES

[1] O. Brudaru, A. Vîlcu, **Genetic Algorithm with accelerating hybrid components for minimal affine cost Hamiltonian circuits**, MIT Press, 1998.

[2] O. Brudaru, A. Vîlcu, **Genetic Algorithm with Accelerating Hybrid Components for Affine Cost Hamiltonian Circuits**, ICPR-16, 16th International Conference on Production Research, Prague, 2001.

[3] R. A. Howard, **Dynamic programming and Markov processes**, MIT, John Wiley & Sons, New York, 1960.

[4] C. Nilsson, **Heuristics for the Traveling Salesman Problem**, MIT Press, 2004

"Gheorghe Asachi" Technical University of Iaşi
Department of Engineering and Management
63 Dimitrie Mangeron, Iasi, ROMANIA
e-mail: avilcu@misp.tuiasi.ro