Centre Of Mass Selection Operator Based Meta-Heuristic For Unbounded Knapsack Problem

D.Venkatesan, K.Kannan and S. Raja Balachandar

Abstract—In this paper a new Genetic Algorithm based on a heuristic operator and Centre of Mass selection operator (CMGA) is designed for the unbounded knapsack problem(UKP), which is NP-Hard combinatorial optimization problem. The proposed genetic algorithm is based on a heuristic operator, which utilizes problem specific knowledge. This center of mass operator when combined with other Genetic Operators forms a competitive algorithm to the existing ones. Computational results show that the proposed algorithm is capable of obtaining high quality solutions for problems of standard randomly generated knapsack instances. Comparative study of CMGA with simple GA in terms of results for unbounded knapsack instances of size up to 200 show the superiority of CMGA. Thus CMGA is an efficient tool of solving UKP and this algorithm is competitive with other Genetic Algorithms also.

Keywords—Genetic Algorithm; Unbounded Knapsack Problem; Combinatorial Optimization; Meta-Heuristic; Center of Mass

I. Introduction

Knapsack problem is a well known and well-studied problem in combinatorial optimization, however polynomial time algorithm for many cases have not been tried yet. It has a wide range of applications, ranging from network planning, network routing, parallel scheduling, to budgeting. As an application, it is worth mentioning that the unbounded knapsack problem(UKP) often arises in cargo loading and packing . Suppose we need to load a vessel of capacity c with a cargo composed of different types of items with each item of type i having weight w_i and value p_i. The problem is how to load the vessel with the most valuable cargo [15,5]. The unbounded knapsack problem is NP-hard [15,10]. It may be formulated as follows: a knapsack of capacity c is given, into which we may put n types of objects. Each object of type i has a profit p_i and a weight w_i (without loss of generality, we assume that w_i , p_i , c and n are all positive integers, and an unbounded number of items of each type is available). Determine the number x_i of i th type objects that maximize total profit without exceeding capacity, i.e.

$$maximize \sum_{i=1}^{n} p_i x_i \qquad (1)$$

subject to

D.Venkatesan is with the School of Computing, SASTRA University,Thanjavur,INDIA, e-mail:venkatgowri@cse.sastra.edu

K.Kannan is with the Department of Mathematics, SASTRA University,Thanjavur,INDIA, e-mail: kkannan@maths.sastra.edu

S.Raja Balachandar is with the Department of Mathematics, SASTRA University,Thanjavur,INDIA, e-mail: srbala@maths.sastra.edu

$$\sum_{i=1}^{n} w_i x_i \leq c \qquad (2)$$

$$x_i \geq 0 \ are \ integers, i = 1,...,n \qquad (3)$$

Two classical approaches for solving this problem exactly are Branch and Bound [20] and Dynamic Programming [1]. Both of these exact algorithms have a running time that is bounded by an exponential function of the length of input data. Thus it is very difficult to obtain the exact solutions to many large- scale knapsack instances which come from practical applications [5]. Hence for those large-scale instances, it has to rely on heuristic algorithms to obtain the near optimal solutions to them [2, 9,12, 14,16, 19, 18].

Among those heuristic algorithms for knapsack problem, Genetic Algorithm is found to be an effective method to solve the knapsack instances approximately. Ken-li Li [13] presented a problem-specific genetic algorithm for solving the UKP. They have solved the UKP by transforming it into 0-1 knapsack problem. In this paper, we have also used the problem-specific heuristic operator (described in section 3.3) to convert an infeasible solution into a feasible one, and we have directly solved the UKP without transforming it into 0-1 knapsack problem. We have used a new operator called Center of Mass Selection operator [17] (described in section 3.2), One Point Crossover and the Mutation operators to solve the UKP. The proposed algorithm is capable of obtaining more exact solutions for various problems.

The rest of this paper is organized as follows. In section II, we presented an introduction to Genetic Algorithm. In section III, a new Genetic Algorithm containing the heuristic operator for solving the UKP is explained. Experimental results are presented in section IV, and concluding remarks are provided in section V.

II. An Introduction to Genetic algorithm

Genetic Algorithms are algorithms for optimization and machine learning based loosely on several features of biological evolution. They require five components [21].
(i)A way of coding solutions to the problem on chromosomes.
(ii) An evaluation function, which returns a rating for each chromosome given to it.
(iii) A way of initializing the population of chromosomes.
(iv) Operators that may be applied to parents when they reproduce to alter their genetic composition. Standard operators are mutation and crossover.

(iv) Parameter settings for the algorithm, the operators, and so forth.

Given these five components, a Genetic Algorithm operates according to the following steps.

Step 1. Initialize the population using the initialization procedure and evaluate each member of the initial population.

Step 2. Reproduce until a stopping condition is met. Reproduction consists of iterations of the following steps:

a) Choose one or more parents to reproduce. Selection is stochastic, but the individuals with the highest evaluations are favored in the selection.

b) Choose a genetic operator and apply it to the parents.

c) Evaluate the children and accumulate them into a generation. After accumulating enough individuals, insert them into the population, replacing the worst current members of the population.

When the components of the Genetic Algorithm are chosen appropriately, the reproduction process will continuously improve the population, converging finally to solutions close to a global optimum. Genetic Algorithms can efficiently search large and complex (i.e., possessing many local optima) spaces to find nearly global optima. A more comprehensive overview of Genetic Algorithm can be found in [2,3,4,11].

A. A brief survey of GA based applications, wherein the definition of basic operators need revision

Goldberg[11]suggested a modified crossover operator called Partially Matched Crossover (PMX) for solving the Traveling salesman problem and the other crossover operators are Order Crossover(OX) and Cycle Crossover(CX) for the Traveling salesman problem to maintain the tour. He also defined some new mutation operators, namely, Inversion Mutation, Insertion Mutation, Displacement Mutation, Reciprocal Exchange Mutation and Shift Mutation for solving various problems. The standard selection operators such as roulette wheel, rank, tournament, etc., have been used by various researchers for varied applications. In this paper we have used a new selection operator based on center of mass (described in section 3.2) for solving UKP.

B. A brief survey of problem-specific operators used with GA applications

P.C.Chu and J.E.Beasley [7] used a problem specific heuristic operator to improve a GA based solution for solving generalized assignment problem. It includes two local improvements. The first one attempts to improve feasibility of a child by reassigning jobs from overly-capacitated agents to less-capacitated agents and the second one attempts to improve the cost of the child by reassigning jobs to agents with lower costs. The heuristic operator has a complexity of $O(m^2 n)$. P.C.Chu and J.E.Beasely [8] used a heuristic operator, like what we have used in our paper for solving multi-dimensional knapsack problem.

III. THE CENTER OF MASS SELECTION OPERATOR BASED GENETIC ALGORITHM FOR SOLVING UKP (CMGA)

.

A. Representation and Fitness function

The binary coded Genetic algorithm is used for solving the UKP shown below. The initial population of size 100 has been chosen randomly, with each chromosome represented by the total length l , which represents the sum of total number of bits required for the upper bound of each variable i. Let n be the total number of variables.

Step 1: (initialize) construct P random candidate solutions.

Step 2: (stopping criteria) if maximum number of iteration is reached go to Step 10

Step 3: choose the best solution among the solutions

Step 4: Center of Mass = best solution

Step 5: (selection) apply Center of Mass selection operation

Step 6: (crossover) apply single-point crossover operation

Step 7: (Mutation) apply Mutation operation

Step 8: (feasibility operator) Apply the feasibility operator to convert the infeasible solutions into feasible ones.

Step 9: go to Step 2

Step 10: display the best solution.

B. Parent selection, Crossover and Mutation

We have used our own selection operator called Center of Mass Selection operator. This operator uses the neighborhood of the best string in the population for selection. It quickly identifies the local optimum within the current population. Here, the best fit individual (individual with the highest objective function value) is taken as the Center of Mass x^c and the new individuals x^{new} are generated around the Center of Mass, by using equation (4).

$$x_i^{new} = x_i^c + \frac{l_i r}{k} \quad \forall i \qquad (4)$$

Where x_i^c is the ith variable in the Center of Mass, l_i is the upper bound of x_i, r is the normalized random number and k is the iteration number. For example, consider the 5-variable UKP. Let the upper bounds for the 5 variables be 15, 18,13,23,13. The number of binary digits for each variable is: 4,5,4,5,4. So, the total length $l = 22$. we present conversion of old chromosome into new chromosome in TABLE I. Assume that the following chromosome (individual) is the center-of-mass. $x^c = 1101001010001101010011$

TABLE I
CONVERSION OF x^c INTO x^{new}

x^c (binary)	1101001010001101010011
x^c (decimal)	13, 5, 1, 21, 3
Iteration k	10
x^{new}(decimal)	13+(15*0.7)/10 = 14
	5+(18*0.6)/10 = 6
	1+(13*0.9) /10 = 2
	21+(23*0.1)/10 = 21
	3+(13*0.5)/10 = 4
x^{new} (binary)	1110001100010101010100

So, the new chromosome is created as $x^{new} = $ 1110001100010101010100 Like this the remaining chromosomes will be generated in the population. We have used One Point Crossover operator with the crossover probability of 0.9. It will select two parents and based on a random number it will get the position value, from which the remaining string from the two parents get exchanged. We have also used Mutation

operator with the probability of 0.05 (which gives better results compare to the other values). It will randomly change some of the bits from 0 to 1 or from 1 to 0.

C. The heuristic Operator

During the evaluation of the fitness functions of the population, we use this problem specific heuristic operator to transform the infeasible solutions into a feasible one. It is presented below. It consists of two phases. The first phase (called Drop) examine each variable in increasing order and changes the variable from one to zero if feasibility is violated. The second phase (called Add) reverses the process by examining each variable in decreasing order and changes the variables from zero to one as long as feasibility is not violated. The aim of the Drop phase is to obtain feasible solution from an infeasible solution, while the Add phase seeks to improve the fitness of a feasible solution.

Let R = the accumulated weight of the knapsack S.

1. initialize R $=\sum w_i * S[m]$;
2. for j = 1 to l do /* Drop phase */
if (S[j] = 1) and (R>c) then
S[j] = 0; R = R - w_j
end if
end for
3. for j = l to 1 do /* Add phase */
if (S[j] = 0) and ($R < c$) then
S[j] = 1; R = R + w_j
end if
end for.
4. end

D. example

For example, consider the same 5-variable UKP, with the constraint equation is given by R = 77x1 + 65x2 + 88 x3 + 52x4 + 90x5 $\leq$ 1197.

Consider x^{new} = 1110001100010101010100

If we substitute this in the constraint equation above, we get R = 3096 which is greater than 1197. so, the solution is infeasible. To convert it into feasible one, we apply the heuristic operator as follows: Drop phase: It replaces 1 by 0 from left to right as long as it is infeasible. The modified string is 0000000000000001010100 Add phase: It replaces 0 by 1 from right to left as long as it satisfies the constraint. The resultant string is 0000000000001111111111 which is a feasible solution. In terms of computational complexity both step. 2 and step. 3 require at most O(l), which is relatively small.

IV. EXPERIMENTAL RESULTS

This algorithm has been tested for solving UKP (which is NP hard)with single constraint so as to check its efficiency in the context of vast search space. We generated the Unbounded Knapsack test problems randomly for various sizes(n), following the procedure given in [6], which is described below. The value of p_j varies from 1 to 1000, w_j varies from 1 to 1000, b_j varies from 1 to 10. The right hand side value c is calculated

by using $0.5 * \sum w_j * b_j$. This algorithm was implemented in Visual Basic 6.0 and tested in Intel Core2 Duo , 1.60 GHz. Processor and 1.0 GB RAM running under Windows XP.

A. Solution quality

TABLE II summarizes the results for each of the solved problem. The Columns in the table are furnished as follows:
n : The number of variables
CPLEX: CPLEX solutions(optimum)
Opt : The number of trails out of 10 in which CMGA found the optimum solutions.

Mean, Min, Max : The Mean, Minimum and Maximum Objective values returned in the 10 trails (the val column) and the respective relative deviation percentages below the optimal value (in dev column). The CMGA found optimal solutions in at least one of the 10 trails for all the 8 tested problems. For 4 (50 percentage) of the problems the CMGA found the optimal solution in every trails. Only in two cases, at n = 40 it returns a solution 5.763 percentage below the optimal and at n = 150 it returns a solution 3.641percentage below the optimal. The CGMA to return best solutions for all sizes of problems.

TABLE II
CMGA RESULTS

n	CPLEX	Opt	Mean		Min		Max	
			Val	dev	Val	dev	Val	dev
10	3293	10	3293	0.0	3293	0.0	3293	0.0
20	6586	10	6586	0.0	6586	0.0	6586	0.0
30	10353	10	10353	0.0	10353	0.0	10353	0.0
40	13708	9	13629	0.576	12918	5.763	13708	0.0
50	16836	10	16836	0.0	16836	0.0	16836	0.0
100	34272	8	34245	0.078	34202	0.204	34272	0.0
150	53224	9	53030.2	0.364	51286	3.641	53224	0.0
200	77319	8	77250	0.089	76939	0.491	77319	0.0

No. of iterations : 3000 No. of runs:1 CPLEX : software

B. Comparison with other Algorithms

In order to bring out the efficiency of the proposed CMGA , the set of generated problems have been solved by Genetic Algorithm(GA) given in [13]. The values of the parameters used in both the algorithm are listed in TABLE III. In TABLE IV, we have compared our CMGA algorithm with the GA described in [13]. In [13], the tournament selection operator is used to solve the UKP. The tournament selection operator picks two random solutions from the initial solutions and the selection is made randomly based on a threshold value of the user's choice. Hence the randomness affects the solution quality. But in this paper, center of mass selection operator preprocesses the data to obtain the best initial solution, by assuming this initial solution as center of mass and generate the population existing around it, as candidate solutions for the next generation. This causes the improvement in the solution quality for successive generations. Simple GA obtained only the near optimum solutions with average deviation of 0.0571 percentage. Where as this algorithm obtained optimum solutions in all the test instances. This is solely because of the center of mass selection operator.

V. CONCLUSION

In this paper a new Genetic Algorithm based on Centre of Mass selection operator for solving the UKP is presented. Our

TABLE III
PARAMETERS SET FOR THE TWO ALGORITHMS

Parameters	GA	CMGA
Population size	100	100
Selection	Tournament	Center of Mass
Crossover	Uniform (0.9)	One point (0.9)
Mutation	0.05	0.05
Max.iterations	3000	3000

TABLE IV
COMPARISON BETWEEN CMGA AND GA

n	CPLEX Optimum Value	GA	dev	CMGA	dev
10	3293	3208	0.026	3293	0
20	6586	6586	0	6586	0
50	16836	15636	0.071	16836	0
100	34272	33256	0.030	34272	0
150	53224	45386	0.147	53224	0
200	77319	71984	0.069	77319	0
Total average			0.0571		0

No. of iterations: 3000 No. of runs: 10 CPLEX: software

approach is different form the approach of Ken and Qing[13] . They have solved the UKP by transforming it into 0-1 knapsack problem, and then by applying simple GA with tournament selection. But our algorithm does not necessitate the transformation process which causes the reduction of memory space. Instead a new operator called Center of Mass Selection operator has been used along with One Point Crossover and the Mutation operators to solve the UKP. The proposed algorithm is capable of obtaining better solutions for various problems and this method gives quality solutions than the GA described in [13]. We have used our new algorithm to solve only single constraint NP hard problem up to problem size 200. However this algorithm needs to be tested for solving problems of larger size. The proposed algorithm can be used to solve multi constrained optimization problems and also for multi objective optimization problems, which is presently ongoing.

REFERENCES

[1] Andonov R, Poirriez V, Rajopadhye S, Unbounded knapsack problem: Dynamic Programming revisited, European Journal of Operation Research, Vol. 123, pp.394-407, 2000.

[2] Back T. Fogel D B, Michalewiez Z (eds.), Handbook of Evolutionary Computation, Oxford University Press, 1975.

[3] D.Beasley, D.R.Bull, and R.R.Martin ,An overview of genetic algorithms: Part I. fundamentals, University computing , 15, 58-69, 1993.

[4] D. Beasley, D.R.Bull, and R.R.Martin, An overview of genetic algorithms: Part II. Research topics, University computing ,15,170-181, 1993.

[5] Bellman R. and Dreyfus.S.E., Applied Dynamic Programming, Princeton University Press, Princeton, NJ, 27-31 , 1962.

[6] Chanin Srisuwannapa, Peerayuth Charnsethikul, An Exact Algorithm for the Unbounded Knapsack problem with Minimizing Maximum Processing Time, Journal of Computer Science 3 (3): 138-143, 2007.

[7] P.C.Chu and J.E.Beasely, A Genetic Algorithm for the Generalized Assignment Problem, Computer Ops. Res. , vol. 24, No.1, 17-23, 1997..

[8] P.C.Chu and J.E.Beasely, A Genetic Algorithm for the Multi-dimensional Knapsack problem, Journal of Heuristics, Vol. 4, 63-86,1998.

[9] Dudzinski.K, A note on dominance relation in unbounded knapsack problems, Operation Research Letters, 10(7), 417-419, 1991.

[10] Garey M R. Johnson D S, Computers and intractability, A guide to theory of NP-Completeness, Freeman and Co., San Francisco, 1979.

[11] D.E. Goldberg, Genetic Algorithms in search, optimization and machine learning, Addition Wesley, Reading, MA,1989.

[12] Hans Kellerer, Ulrich Pferschy, David Pisinger, Knapsack Problems, Springer - Verlag , 2003.

[13] Ken-li Li, Guang-ming Dal, Qing-hua Li, A Genetic Algorithm for the Unbounded Knapsack Problem, Proceedings of the Second International conference on Machine Learning and Cybernetics, Xi'an, IEEE, 2-5 November 2003.

[14] Kulanoot Araya , Algorithms for some hard knapsack problems , PhD Thesis , Curtin University of Technology, 2000.

[15] 15. Martello S Toth P, Knapsack problems: Algorithms and Computer implementation, Wiley, New York, 1990.

[16] Martello.S, toth.P., An exact algorithm for large unbounded knapsack problems, Operation Research letters, 9(1), 15-20, 1990.

[17] Osman K.Erol, Ibrahim Eksin, A new optimization method: Big Bang-Big Crunch, Advances in Engineering Software 37, 106-111, 2006.

[18] Poirriez, V., Yanev, N., Andonov, R., A hybrid algorithm for the unbounded knapsack problem, Discrete Optimization, 6(1),110-124, 2009.

[19] Rung-Ching Chen, Cheng-Huei Jian,Yung-Fa Huang, Solving Unbounded Knapsack Problem using an Adaptive Genetic Algorithm with Elitism Strategy, International Journal of Smart Home, Vol. 2, No. 2, 139-150, 2008.

[20] Shih.w, A branch and bound method for the multi constraint 0-1 knapsack problem, J.Oper.Res.Soc., 30, 369-378, 1979.

[21] D.Venkatesan, K.Kannan, R.Saravanan, A genetic algorithm-based artificial neural network model for The optimization of machining processes, Neural Computing and Applications, 18,135-140, 2009.