Neural, Genetic, And Neurogenetic Approaches For Solving The 0-1 Multidimensional Knapsack Problem

Jason Deane, Ph.D., Virginia Tech, USA
Anurag Agarwal, Ph.D., University of South Florida, USA

ABSTRACT

The multi-dimensional knapsack problem (MDKP) is a well-studied problem in Decision Sciences. The problem's NP-Hard nature prevents the successful application of exact procedures such as branch and bound, implicit enumeration and dynamic programming for larger problems. As a result, various approximate solution approaches, such as the relaxation approaches, heuristic and metaheuristic approaches have been developed and applied effectively to this problem. In this study, we propose a Neural approach, a Genetic Algorithms approach and a Neurogenetic approach, which is a hybrid of the Neural and the Genetic Algorithms approach. The Neural approach is essentially a problem-space based non-deterministic local-search algorithm. In the Genetic Algorithms approach we propose a new way of generating initial population. In the Neurogenetic approach, we show that the Neural and Genetic iterations, when interleaved appropriately, can complement each other and provide better solutions than either the Neural or the Genetic approach alone. Within the overall search, the Genetic approach provides diversification while the Neural provides intensification. We demonstrate the effectiveness of our proposed approaches through an empirical study performed on several sets of benchmark problems commonly used in the literature.

Keywords: Neural Networks; Integer Optimization; Multidimensional Knapsack Problem; Genetic Algorithms; NeuroGenetic Approach.

1. INTRODUCTION

The multi-dimensional knapsack problem (MDKP) has been well researched since the 1950s and continues to receive attention due to its wide applicability. Many real-world problems including the capital budgeting problem (Salkin and Kluyver, 1975), the cutting stock problem (Gilmore and Gomery, 1966), the cargo loading problem (Shih, 1979), the set covering problem (Bitran and Hax, 1981), production planning problem (Camargo et. al., 2012) the distributed processor and database allocation problem (Gavish and Pirkul, 1982), the Stigler diet problem (Lancaster, 1992) and many others (Salkin and Kluyver, 1975) can all be formulated as a MDKP. Additionally, the MDKP often appears as a sub problem in a variety of more difficult problems such as the generalized assignment problem. Any incremental solution improvement would be beneficial to many applications.

In this problem, we are given a set $T = \{1 \ldots n\}$ of n items and a set $R = \{1 \ldots m\}$ of m resources. Each item $i \in T$ has a value c_i and a vector of non-negative constraints such as weight or size A_{ij}, which represent the amount of resource j consumed by item i. In addition, each resource $j \in R$ has a capacity b_j. The objective is to identify the subset of items which, when added to the knapsack, maximizes the total knapsack value without violating any of the resource constraints.

Formally, the problem can be stated as:

$$\text{Max} \quad \sum_{i=1}^{n} c_i x_i \tag{1}$$

$$\text{st:} \quad \sum_{i=1}^{n} A_{ij} x_i \le b_j, \quad j = 1,\ldots,m$$

$$x_i \in \{0,1\}, \quad i = 1,\ldots,n$$

This knapsack problem belongs to the class of NP-Hard problems (Garey and Johnson, 1979). Like most NP-Hard optimization problems, both exact and approximate solution techniques have been developed for the MDKP. Existing exact solution techniques are primarily based on the branch-and-bound method (Shih, 1979; Martello and Toth, 1981; Gavish and Pirkul, 1985 and Fukunaga, 2011) and the dynamic programming method (Gilmore and Gomery, 1966, Weingartner and Ness, 1967, Nemhauser and Ullmann, 1969 and Volgenant and Zoon, 1990). These techniques work well on small problems, but due to their inherent time complexity and the curse of dimensionality, they are not very practical for larger problem instances. To solve larger instances of the problem, many heuristic approaches, relaxation approaches and metaheuristic approaches have been developed.

Senju and Toyoda (1968) developed a pseudo-utility based heuristic. Kochenberger et. al. (1974) also developed a greedy heuristic based on a pseudo-utility ratio, but with a difference. Both these heuristics are still quite popular. Another greedy heuristic was proposed by Hillier (1969). Other heuristics can be found in (Egeblad and Pisinger, 2009; Boussier et. al., 2010; Archetti et. al., 2010 and Hill et. al., 2012). Various relaxation approaches have also been proposed, such as the combined dual-gradient and greedy method (Loulou and Michaelides,1979), the combined dual-gradient and Lagrangean-relaxation method (Magazine and Oguz, 1984), the surrogate-duality method (Pirkul, 1987), the primal-based parametric approach (Lee and Guignard, 1988), the surrogate-relaxation technique (Freville and G. Plateau, 1997) have also been applied to the MDKP. For details about these techniques, see Lin, 1998.

Many metaheuristic approaches have also been applied to the MDKP. Tabu search is used in (Dammeyer and Voss, 1993) and (Aboudi and K. Jornsten, 1994). Population search techniques such as genetic algorithms are applied in (Chu and J. Beasley, 1998). Other metaheuristic approaches commonly applied to various optimization problems include simulated annealing (Cho and Kim, 1997), ant-colony optimization (Merkle et al., 2002), GRASP (Bard and Feo, 1996) and scatter search (Debels et. al., 2006) and harmony search approach (Zou and Gao, 2011). These metaheuristics are able to find improved solutions by searching a larger portion of the solution space. In general, the longer one runs these metaheuristics, the better the solutions tend to be, although the improvement reduces asymptotically over time. It is up to the user to provide an acceptable stopping criteria. In general, genetic algorithm approaches tend to outperform other approaches.

In this paper, we propose (i) a Neural approach which is based on the principles of Neural Networks, (ii) A Genetic Algorithms approach, and (iii) a Neurogenetic approach which combines the Neural approach and a Genetic Algorithms approach. In the next section we describe the Neural approach. In the third section we will describe the Genetic approach and the in fourth section we will describe the Neurogenetic approach. Empirical results will be discussed in the fifth section.

2. THE NEURAL APPROACH

The Neural approach we propose in this paper is similar to the augmented neural networks approach that has been applied to the task-scheduling problem (Agarwal et. al, 2003, 2006) and Open Shop scheduling problem (Colak and Agarwal, 2005). This approach is a combination of the basic heuristic approach and the neural-network approach. A heuristic typically makes use of some problem-specific parameter(s). For example for the MDKP a heuristic might use the value of the items c_i as the parameter. In the neural approach we use a weight vector, W = $\{w_1, w_2, \ldots, w_n\}$ with initial values of 1 for each of the elements (w_i). We multiply the weight vector with the vector of problem parameters, C = $\{c_1, c_2, \ldots, c_n\}$ used in the heuristic to get a weighted parameter vector WC = $\{w_1 c_1, w_2 c_2, \ldots, w_n c_n\}$. We use this weighted parameter vector (WC) instead of the original parameter vector, C, in the heuristic.

The neural approach is an iterative approach in which each iteration is performed using a heuristic. Since the weight vector is a unit-vector initially, the weighted parameter vector is identical to the original parameter vector (W = WC for the first iteration), so the heuristic solution in the first iteration using the weighted parameter vector produces the same solution as would the original parameter vector. After each iteration, the weight vector is modified using what is called a learning rule. Each subsequent iteration, thus, produces a different solution since the parameter vector is different for each iteration. This approach is very similar to the learning approach used in artificial neural networks. Essentially, this is a local-search approach in which solutions in the vicinity of the heuristic solution are produced.

The Learning Rule

The method by which we modify the weight vector neural network weights is called the learning rule. It is basically a weight modification formula. We employ the following weight modification formulas:

$$w_{i(k+1)} = w_{i(k)} + \alpha * c_i * \varepsilon_{(k)}, \quad \forall i \in T \tag{25}$$

where $\varepsilon_{(k)}$ is the error in the k^{th} iteration.
α is the learning rate.

Heuristics used with the Neural Approach

As explained earlier, in each iteration of the neural approach a heuristic is used to obtain a solution. So whenever a neural approach is used, an appropriate heuristic is selected. The heuristic provides the first solution and then subsequent iterations provide solutions in the neighborhood of the first solution. If the heuristic, and therefore the first solution is produced in a good neighborhood in the search space, the subsequent iterations will provide a local optimal solution in this neighborhood. We apply a modified version of the Senju and Toyoda (1968) and the Kochenberger et al. (1974) heuristics. In the Senju and Toyoda (ST) heuristic, all objects are first added to the knapsack and subsequently objects with the lowest pseudo-utility ratios are removed one at a time until a feasible solution is achieved. In the Kochenberger, McCarl and Wyman (KMW) heuristic, the object with the highest pseudo-utility ratio is added next to the knapsack next until the knapsack is full.

We propose a modification to these heuristics. In the KMW heuristic, all objects are first sorted on pseudo-utility ratio (in descending order) and objects are added to the knapsack starting with the top of the sorted list. The heuristic stops when it is infeasible for the next object in the list to fit in the knapsack. The modification we propose is in the stopping criteria. We argue that stopping at the first infeasible object in the list might be stopping too prematurely because there might be some objects down the list that might be feasible. Even if we were to add one more object, we will improve the solution. So we propose that we exhaust the entire list to see if any object fits. In the ST heuristic, similarly, first all objects are assumed added to the knapsack, even though it produces an infeasible knapsack. The objects are sorted on the pseudo-utility ratio in ascending order, and objects are removed from the knapsack from the top of this list till the knapsack becomes feasible. We propose that after a feasible solution is achieved, instead of stopping, the algorithm should re-evaluate the objects that have been dropped from the knapsack to see if any of them can be added back to the knapsack while maintaining feasibility. Although the refinement requires more computation time, the computational complexity remains the same at *O(nm)*. Empirical analysis revealed that the improvement due to this refinement was significant in relation to the small amount of incremental processing time required. We call the two modified heuristics KMW-M and ST-M.

3. GENETIC ALGORITHMS APPROACH

When solving the knapsack problem using a genetic algorithm, the chromosome string consists of 0s and 1s, with 0s representing absence and 1s representing presence of an item in the knapsack. The length of the chromosome string is equal to the number of items in the given problem. Chu and Beasley, 1998, have proposed a very effective genetic algorithm approach. A Typical GA algorithm is shown in Figure 1.

Generate an initial population
Evaluate the fitness of individuals in the initial population
Repeat
    Select parents for crossover
    Crossover parents to produce an offspring
    Repair offspring (i.e. make it feasible)
    Evaluate offspring
    If offspring is better than any individual in population replace that individual with the offspring.
Until Some stopping criteria is met.

Figure 1: A GA Algorithm

Depending on the problem, the crossover operator may not necessarily produce an offspring that represents a feasible solution. In the traveling salesman problem, for example, the crossover always produces a feasible solution, because any permutation of nodes is a feasible tour. In the knapsack problem, the offspring can easily represent either an infeasible solution or a solution with all non-binding constraints. In each case a repair operator is needed to either eliminate the infeasibility or to bring the solution closer to a boundary solution. The repair operator thus involves decisions about which items to remove (from an infeasible solution) or which items to add (for a non-boundary solution).

Chu and Beasley (1998) proposed a repair operator within which for each item *i*, a ratio $u_i = c_i / \sum_{j=1}^{m} \omega_j A_{ij}$ is calculated. If the solution is infeasible, items are arranged in increasing order of u_i and items are dropped in increasing order of their u_i value, until the solution becomes feasible. If there is room for more items, items are added in the decreasing order of u_i. This idea is similar to the underlying idea behind the KMW and ST heuristics discussed in Section 2. We propose a repair operator which instead of using the ratio u_i, uses the values of the items in the LP solution of the relaxed version of the problem. Items are added in increasing order and dropped in decreasing order of the value of each variable.

We also make use of the LP-relaxed solution in the development of an initial population for the GA. Items are arranged in a decreasing order of LP-solution values. Items with highest LP-solution values are considered first, but added randomly if the random number generated is greater than 0.5. When infeasibility reaches, the item causing the infeasibility is not added. In the remainder of the paper, we will refer to the GA with the proposed refinements as the AD-GA, short for Agarwal-Deane Genetic Algorithm. We will refer to Chu and Beasley's GA as CB-GA.

4. THE NEUROGENETIC APPROACH

Based on our empirical experience with the Neural and GA approaches we made the following observations:

- The Neural approach quickly improves upon the starting solution, but reaches its pinnacle very quickly; most of the improvement occurs in the first 25 or so iterations with very little improvement thereafter.
- GA does not improve as quickly as the Neural approach at first, but keeps improving at a slow but steady rate for a much longer time. Of course the rate of improvement slows down asymptotically, but not nearly as abruptly as with the Neural approach.

These observations suggest that the Neural and the GA search approaches are potentially complementary and that if combined in a manner that takes advantage of each techniques' strengths, a hybrid approach between them may work better than either one independently. Motivated by these observations, we propose an interleaved approach within which after a few iterations of GA, the best GA solution is fed to the Neural approach, which tries to improve upon this solution for about 25 iterations and feeds its best solution back to the GA population. This iterative process continues until a pre-defined stopping criterion is met. We call this interleaved search technique, the Neurogenetic Approach.

5. EMPIRICAL TESTING AND RESULTS

All the proposed methods in this paper were coded in Visual Basic .Net and executed on a Pentium-IV 2.8 GHz machine. For testing purposes, we use data sets available from OR-library. There were three sets of problems. The first set consisting of 7 problem instances and a second set of 48 instances for a total of 55 instances. These 55 instances have served as benchmark problems for several past studies on MDKP. These instances are relatively small, with *m* ranging from 2 to 30 and *n* ranging from 6 to 105. Optimal values of these instances are also known through exact methods. The third problem set consists of 270 problems, first created by Chu and Beasley (1998). The 270 problems by Chu and Beasley are larger problems, with *m* ranging from 5 to 30 and *n* ranging from 100 to 500. Due to the large size of these problems, their optimal solutions are yet unknown. These 270 problems are divided into nine sets of 30 problems each. For details on how the problems were generated.

5.1 Results for 55 standard instances

Table 1 summarizes the results for the 55 benchmark instances, divided into two datasets, dataset-1 of seven instances while dataset-2 of 48 instances. Since the optimal solutions to these problems are known, all results are reported in terms of average percent deviation from the optimal. Single-pass heuristics KMW solve no problems to optimality and give an average deviation from the optimum of 8.06% for dataset-1 and 4.68% for dataset-2. The proposed modified heuristic KMW-M solved 1 instance from dataset-1 and 8 from dataset-2 to optimality with average deviation of 2.14% and 0.60% for the two datasets respectively. The proposed refinement to the KMW thus provided a significant improvement (from 8.06% deviation down to 2.14% deviation). ST and ST-W heuristics showed similar results. While ST found no optimum solutions for dataset-1 and three for dataset-2 with deviations of 9.19% and 8.5% for the two datasets respectively, ST-M provided one optimal solution for dataset-1 and six for dataset-2 with deviations of 3.41% and 0.61% respectively. The CPU time taken was negligible for both heuristics – of the order of 10^{-3}.

Table 1
Results by Various Techniques for the 55 Benchmark Problems

		Dataset-1 (7 instances)			Dataset-2 (48 instances)		
	Iterations	Percent Dev. from Optimum	Number of Optimum Solutions	Avg. CPU time (seconds)	Percent Dev. from Optimum	Number of Optimum Solutions	Avg. CPU time (seconds)
Single-Pass, KMW	1	8.062	0	0.001	4.681	0	0.0034
Single-Pass ST	1	9.187	0	0.001	8.501	3	0.0021
S-Pass, KMW-M	1	2.14	1	0.0134	0.60	8	0.0045
S-Pass, ST-M	1	3.41	1	0.0028	0.61	6	0.0039
Neural, KMW-M	100	0.76	3	0.10	0.19	23	1.29
	1K	0.34	5	1.80	0.14	31	4.83
Neural, ST-M	100	0.51	3	0.09	0.43	25	0.69
	1K	0.31	4	1.48	0.27	37	6.51
CB-GA	1K	0.20	5	0.56	0.38	24	0.85
	5K	0.11	5	2.51	0.15	41	2.19
	10K	0.09	5	4.73	0.15	44	3.73
AD-GA	1K	0.11	5	0.44	0.34	35	0.96
	5K	0.00	7	3.08	0.12	45	3.12
	10K	0.00	7	6.11	0.11	46	6.01
Neurogenetic, CB	1K	0.11	5	0.71	0.26	29	1.33
	5K	0.09	5	2.63	0.15	39	3.28
	10K	0.10	5	5.18	0.01	44	7.11
Neurogenetic, AD	1K	0.08	5	0.74	0.14	39	1.39
	5K	0.00	7	3.06	0.13	44	3.92
	10K	0.00	7	6.23	0.00	48	7.86

The Neural approach, in 100 unique evaluations, in conjunction with KMW-M, solved 3 out of 7 instances from dataset-1 and 23 out of 48 instances from dataset-2 to optimality. In 1000 unique evaluations, the Neural (with KMW-M) approach solved 5 from dataset-1 and 31 from dataset-2 to optimality. The deviations reduced from

2.14% for KMW-M down to 0.76% for 100 evaluations and 0.34% for 1000 evaluations for dataset-1. However, the average CPU time taken per problem for 1000 evaluations was quite significant – around 5 seconds. Results of Neural approach with ST-M were similar.

We also compared results of the proposed AD-GA approach with the CB-GA approach. AD-GA showed improvements over CB-GA for each dataset, both in terms of reduced deviations and the number of problems solved to optimality. AD-GA was able to find the optimal solutions to all seven problems in 5,000 evaluations, whereas CB-GA could only solve 5 problems to optimality, even after 10,000 evaluations. The AD-GA approach consumed a little extra CPU time compared to CB-GA for the same number of evaluations, but gave better solutions in much fewer evaluations; hence in terms of time taken AD-GA actually performed better – i.e. better results in shorter computing time.

The Neurogenetic approach gave improved results compared to GA-alone, both with AD-GA and CB-GA. For example, for dataset-2, Neurogenetic-CB, in 1,000 evaluations found 29 optimal solutions, compared to only 19 by CB-GA alone. For 10,000 evaluations, although the number of optimal solutions found remained the same, at 44, the average deviation by Neurogenetic-CB reduced to 0.01% down from 0.07% by CB-GA. Similar improvements were observed in Neurogenetic-AD compared with AD-GA alone. For dataset-2, for example, in 1,000 evaluations, Neurogenetic-AD found 39 optimal solutions compared to 23 by AD-GA alone. In 10,000 evaluations, Neurogenetic-AD found all 48 optimal solutions, compared to 42 found by AD-GA alone. The deviation was reduced from 0.15% to 0.00%.

5.2 Results for Chu and Beasley Dataset

Table 2 summarizes the results for the 270 test instances by Chu and Beasley, 1998. Since optimal solutions to these large problems are unknown, the results are reported in terms of average percent deviations from the relaxed LP-based solution. The refinements to KMW and ST heuristics provided some improvements over the original KMW and ST heuristics. For example, KMW-M gave an average deviation of 1.5% compared to 2.1% by KMW. Similarly ST-M reduced the deviation to 1.42% down from 1.76% for ST. Neural approach with KMW-M, in 100 iterations, reduced the deviation to 1.1% and to 1.0% in 1000 iterations. Neural approach with ST-M reduced the deviation to 1.00% in 100 iterations and to 0.99% in 1000 iterations. We observed that Neural approach provided most of its improvements within the first 100 evaluations. No significant improvements were observed going from 100 to 1,000 evaluations. When comparing the results of Neural with CB-GA, we observed that although the Neural approach provided better gaps compared to CB-GA for the first 100 evaluations (1.1% compared to 1.4%), GA-CB continued to improve with more evaluations. CB-GA consumed less CPU for 100,000 iterations than did Neural approach for 1,000 evaluations and CB-GA in 100,000 evaluations reduced the gap to 0.63% compared to Neural approach's gap of 1.0%.

The proposed AD-GA approach showed significant improvement over CB-GA at each level of evaluation, albeit at a slight expense of CPU time (See Figure 3a). The domination of AD-GA over CB-GA was particularly significant on larger problems of size n = 500 (See Figure 4a). For n=100 problems, although AD-GA provided better results, the improvement was not very significant. This could also be due to the fact that for n=100 problems, there is not much room left for improvement.

The effectiveness of the Neurogenetic approach is also demonstrated through these test problems. The Neurogenetic approach in conjunction with CB-GA provided marginal improvement at each level of evaluation (See Figure 2b for all 270 problems and Figure 3b for large-size problems). The gaps with Neurogenetic approach were about 0.02% to 0.03% lower compared to CB-GA alone, for the same number of evaluations. The CPU time consumed, although higher, was not significantly higher – 122 seconds for 100,000 evaluations for Neurogenetic-CB compared to 104 seconds for CB-GA. The Neurogenetic approach in conjunction with AD-GA also gave marginal improvement of over AD-GA alone (See Figure 2c for all 270 problems and Figure 3c for large-size problems). In the Neurogenetic runs, we provided 5 boosts of local improvements by the Neural approach. In each boost, we randomly chose two solutions from the ten best solutions in the GA population and ran 15 iterations on each of those two solutions. If the Neural approach found improved solutions, those solutions were incorporated into the GA population for further GA runs. Neurogenetic-AD, in 100K iterations reduced the overall gap to 0.59%, and in 1000K iterations down to 0.54%. AD-GA alone in 1000K iterations reduced the gap to 0.55%.

Table 3 displays the number of instances for which the obtained solution was better than the best solution found by Chu and Beasley, 1998. Neurogenetic-AD found improved solutions to the most number of instances – 34. In addition, it equaled the C&B results for 132 instances. Overall, for all the runs in our empirical study, we found new bounds for 67 of the 270 problems.

Table 2
Results by Various Techniques for the 270 problems by Chu and Beasley

		Percent Deviation from LP Lower Bound				
	Unique Evaluations	**All 270 Probs**	**n = 500 (90 probs)**	**n = 250 (90 probs)**	**n = 100 (90 probs)**	**Average CPU time (seconds)**
Single-Pass, KMW	1	2.036	0.777	1.635	3.696	0.18
Single-Pass ST	1	1.764	0.706	1.355	3.232	0.16
Single-Pass, KMW-M	1	1.498	0.599	1.228	2.666	0.25
Single-Pass ST-M	1	1.422	0.565	1.061	2.640	0.21
Neural, KMW-M	100	1.105	0.497	0.932	1.887	14.10
	1K	1.001	0.476	0.906	1.869	143.83
Neural, ST-M	100	1.003	0.451	0.815	1.761	11.04
	1K	0.993	0.442	0.798	1.698	108.52
CB-GA	1K	1.401	0.980	1.342	1.880	0.66
	5K	0.827	0.466	0.678	1.337	3.89
	10K	0.744	0.368	0.597	1.268	7.72
	100K	0.628	0.274	0.467	1.143	73.99
	1000K	0.562	0.210	0.391	1.086	801.87
AD-GA	1K	0.809	0.296	0.605	1.526	1.04
	5K	0.665	0.244	0.496	1.256	4.18
	10K	0.658	0.238	0.494	1.241	10.05
	100K	0.598	0.213	0.434	1.147	88.15
	1000K	0.553	0.190	0.386	1.082	994.30
Neurogenetic, CB	1K	1.118	0.828	0.937	1.588	0.76
	5K	0.805	0.446	0.641	1.326	4.43
	10K	0.723	0.355	0.556	1.257	8.64
	100K	0.619	0.257	0.450	1.149	78.05
	1000K	0.559	0.204	0.387	1.085	817.47
Neurogenetic, AD	1K	0.742	0.274	0.555	1.397	1.28
	5K	0.665	0.240	0.499	1.257	4.71
	10K	0.643	0.238	0.482	1.209	11.39
	100K	0.590	0.210	0.423	1.137	99.62
	1000K	0.548	0.186	0.378	1.080	1069.45

Table 3
Number of Instances that Improved upon C&B Solutions

	Unique Evaluations	**Number of Solutions equal to C&B solution**	**Number of solutions Better than C&B solutions**	**Total**
Single-Pass, KMW	1	0	0	0
Single-Pass ST	1	0	0	0
Neural, KMW	0.1K	5	0	5
	1K	6	0	6
Neural, ST	0.1K	2	0	2
	1K	2	0	2
CB-GA	1K	0	0	0
	5K	16	0	16
	10K	32	1	33
	100K	61	3	64
	1000K	121	23	144
AD-GA	1K	5	0	5
	5K	27	3	30
	10K	38	4	42
	100K	62	5	67

	1000K	127	27	154
Neurogenetic, CB	1K	2	0	2
	5K	17	0	17
	10K	34	1	35
	100K	71	6	77
	1000K	125	29	154
Neurogenetic, AD	1K	11	0	11
	5K	23	1	24
	10K	38	2	40
	100K	73	7	80
	1000K	132	34	166

Figure - 2a
GA-CB and GA-AD compared on
Percent Deviation Average for all 270 Problems

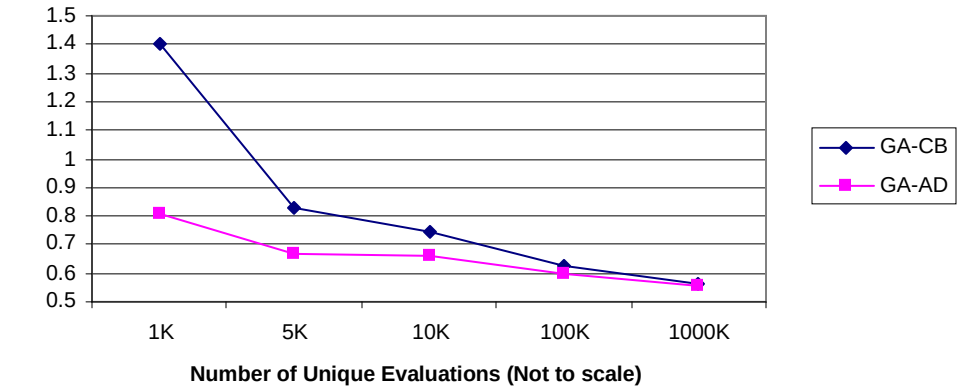


Figure - 2b
GA-CB and NeuroGenetic-CB compared on
Percent Deviation Average for all 270 Problems

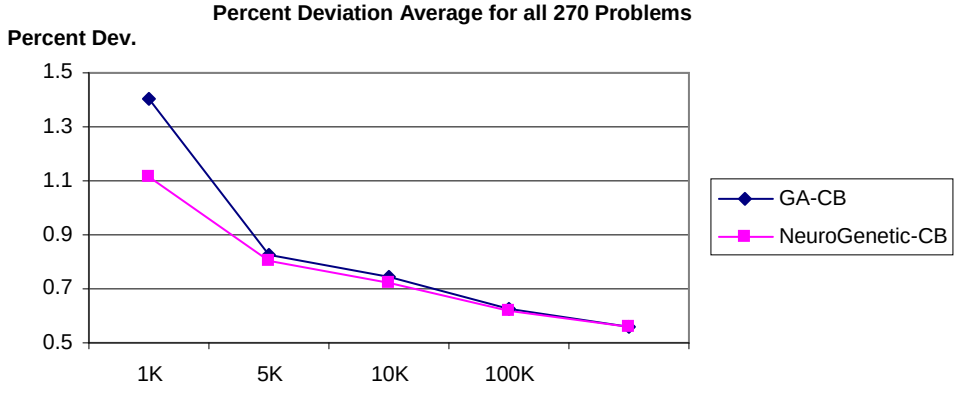


Figure - 2c
GA-AD and NeuroGenetic-AD compared on
Percent Deviation Average for all 270 Problems

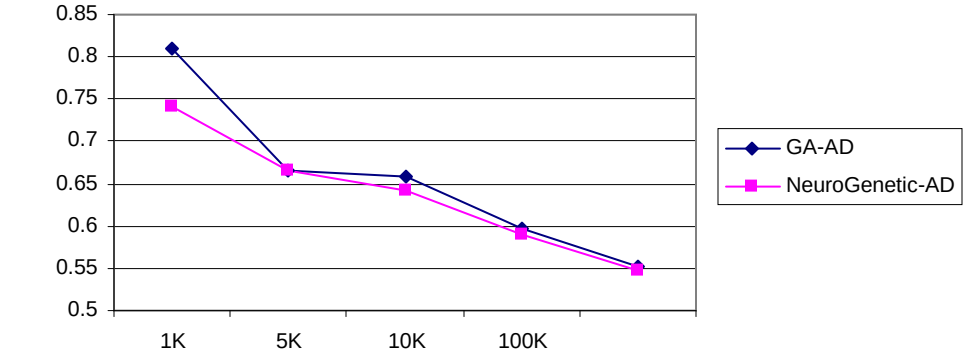
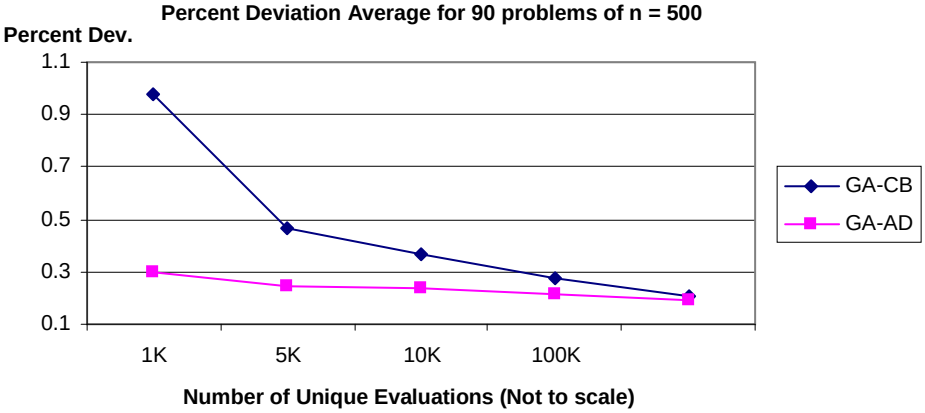
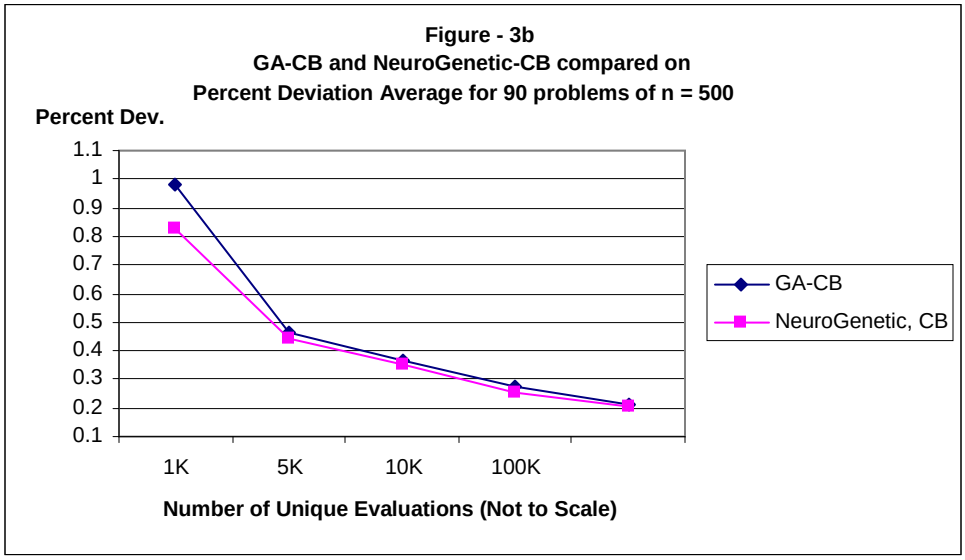
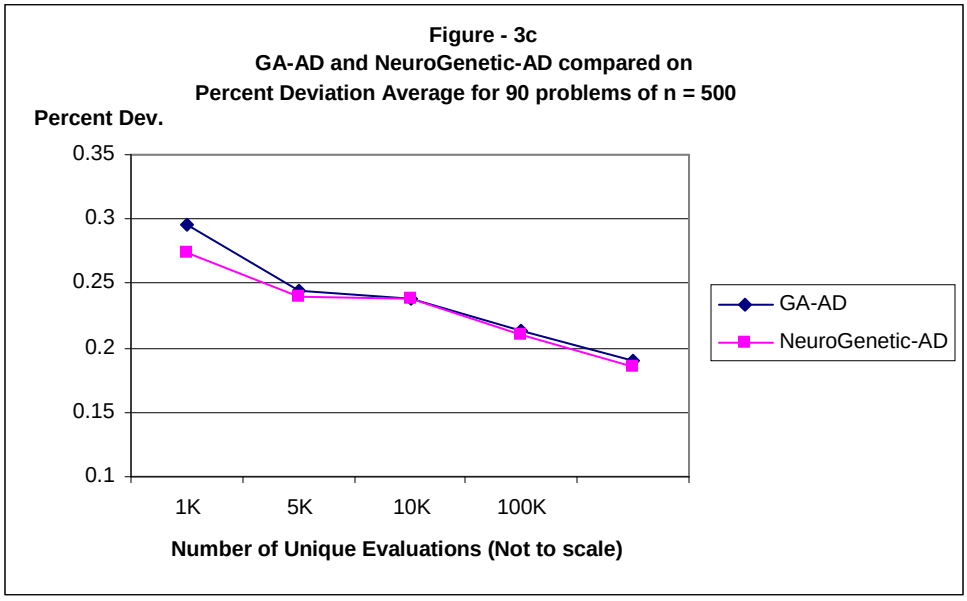


Figure - 3a
GA-CB and GA-AD compared on
Percent Deviation Average for 90 problems of n = 500







6. SUMMARY AND CONCLUSIONS

In this paper we addressed the multi-dimensional knapsack problem (MDKP). We proposed and tested several new methods and ideas for solving the MDKP. First we proposed better stopping criterion for the well-known Kochenberger et al. and Senju and Toyoda greedy heuristics. The proposed criterion provides improved solutions which are closer to the infeasible boundary. Second, we proposed a Neural approach for solving the MDKP and demonstrated its effectiveness compared to the heuristic approaches. In just 100 iterations, deviations from the optimal or relaxed LP-based solutions were reduced by an order of magnitude. Third, we proposed two refinements to the genetic algorithm approach – a new repair operator based on the LP-solution and a new method for generating the chromosomes for the initial population. These refinements improved the performance of the GA. Lastly, we introduced a hybrid technique which interleaves the GA and the Neural approaches. This Neurogenetic approach provides better solutions than either the GA or the Neural approach when applied independently, in almost the same amount of CPU time. We demonstrated the effectiveness of each of these four proposed ideas by testing them on a large set of standard benchmark problems from the MDKP literature. In the process we found new bounds for 67 of the problems.

AUTHOR INFORMATION

Dr. Jason Deane is an Associate Professor of Business Information Technology in the Pamplin College of Business at Virginia Polytechnic Institute & State University. He received a Ph.D. in Decision and Information Sciences from the University of Florida, and an M.B.A. and B.S. in Business Administration from Virginia Tech. His current research interests are in the areas of artificial intelligence, computer aided decision support systems, online marketing, supply chain risk management and information retrieval. E-mail: jdeane1@vt.edu

Dr. Anurag Agarwal is an Associate Professor of Information Systems and Decision Sciences in the College of Business at University of South Florida, Sarasota-Manatee. He earned a Bachelor's degree from Indian Institute of Technology, Roorkee, India, an MBA from University of Wisconsin, La Crosse and a Ph.D. degree in Management Information Systems from The Ohio State University. He teaches a variety of courses in information systems, decision support systems, statistics and operations management. His research interests are in decision support systems, artificial intelligence, operations research and mathematical modeling. He has published his research in a variety of journals. E-mail: agarwala@sar.usf.edu (Corresponding author)

REFERENCES

1. Aboudi, R. and Jornsten, K. (1994). Tabu Search for General Zero-One Integer Programs Using the Pivot and Complement Heuristic. *ORSA Journal on Computing*, 6 82-93.
2. Agarwal A., Pirkul, H. and Jacob, V. (2003). Augmented Neural Networks for Task Scheduling. *European Journal of Operational Research*. 151(3) 481-502.
3. Agarwal, A., Jacob, V.S. and Pirkul, H. (2006). An Improved Augmented Neural-Networks Approach for Scheduling Problems. *INFORMS Journal on Computing*, 18(1) 119-128.
4. Agarwal, A., Colak, S., Jacob, V. and Pirkul, H. (2006). Heuristics and Augmented Neural Networks for Scheduling with Non-Identical Machines. *European Journal of Operational Research*, 175(1) 296-317.
5. Archetti, C., Bertazzi, L. and Speranza, M. G. (2010), Reoptimizing the 0–1 knapsack problem. *Discrete Applied Mathematics.* 158 1879-1887.
6. Bard, J.F., Feo, T.A. and Holland, S. (1996). A GRASP for scheduling printed wiring board assembly. *I.I.E. Transactions*. 28 155-165.
7. Bazgan, C., Hugot, H. and Vanderpooten, D. (2009) Solving efficiently the 0–1 multi-objective knapsack problem. *Computers and Operations Research.* 36 260-279
8. Bitran, G. and Hax, A. (1981). Disaggregation and Resource Allocation using Convex Knapsack Problems with Bounded Variables. *Management Science.* 27 431-441.
9. Boussier, S., Vasquez, M., Vimont, Y., Hanafi, S. and Michelon, P. (2010). A multi-level search strategy for the 0_1 Multidimensional Knapsack Problem. *Discrete Applied Mathematics*, 158 97-109.
10. Camargo, V.C.B., Mattiolli, L. and Toledo, F.M.B. (2012). A knapsack problem as a tool to solve the production planning problem in small foundries. *Computers and Operations Research.* 39 86-92.
11. Cho, J.H. and Kim, Y.D. (1997). A simulated annealing algorithm for resource-constrained project scheduling problems. *Journal of the Operational Research Society*, 48 736–744.
12. Chu, P. and Beasley, J. (1998). A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics*. 4 63-86.
13. Colak, S. and Agarwal, A., (2005). Non-greedy Heuristics and Augmented Neural Networks for the Open-Shop Scheduling Problem. *Naval Research Logistics*. 52 631-644.
14. Dammeyer, F. and Voss, S. (1993). Dynamic Tabu List Management Using Reverse Elimination Method. *Annals of Operations Research.* 41 31-46.
15. Debels, D., Reyck, B.D., Leus, R. and Vanhoucke, M. (2006). A Hybrid Scatter Search / Electromagnetism Meta-Heuristic for Project Scheduling. *European Journal of Operational Research*. 169 (2) 638 -653.
16. Egeblad, J. and Pisinger, D. (2009). Heuristic approaches for the two- and three-dimensional knapsack packing problem. *Computers and Operations Research.* 36 1026-1049.
17. Freville, A. and Plateau, G. (1997). The 0-1 Bidimensional Knapsack Problem: Toward and Efficient High-Level Primitive Tool. *Journal of Heuristics.* 2 147-167.
18. Fukunaga, A.S. (2011). A branch-and-bound algorithm for hard multiple knapsack problems. *Annals of Operations Research*, 184 97-119.

19. Garey, M, and Johnson, D. (1979). Computers and Intractability. A Guide to the Theory of NP-Completeness. *A Series of Books in the Mathematical Sciences*, ed. V. Klee., New York: W.H. Freeman and Company.
20. Gavish, B. and Pirkul, H. (1982). Allocation of Databases and Processors in a Distributed Computing System. *Management of Distributed Data Processing* Ed. J.Akota, North Holland Publishing Company, 215-231.
21. Gavish, B. and Pirkul, H. (1985). Zero-One Integer Programs with Few Constraints – Efficient Branch and Bound Algorithms. *European Journal of Operational Research.* 22 35-43.
22. Gilmore, P. and Gomery, R. (1966). The Theory and Computation of Knapsack Functions. *Operations Research.* 14 1045-1074.
23. Hill, R.R., Cho, Y, K. and Moore, J.T. (2012). Problem reduction heuristic for the 0–1 multidimensional knapsack problem. *Computers and Operations Research*. 39 19-26.
24. Hillier, F., (1969). Efficient Heuristic Procedures for Integer Linear Programming with and Interior. *Operations Research*, 17 600-637.
25. Kochenberger, G., McCarl, B. and Wyman, F. (1974). A Heuristic for General Integer Programming. *Decision Sciences.* 5 36-44.
26. Lancaster, L. (1992). The History of the Application of Mathematical Programming to Menu Planning. *European Journal of Operational Research.* 57 339-347.
27. Lee, J. and Guignard, M. (1988). An Approximate Algorithm for Multidimensional Zero-One Knapsack Problems – a Parametric Approach. *Management Science.* 34 402-410.
28. Lin, E. (1998). A Bibliographical Survey on Some Well-Known Non-Standard Knapsack Problems. *Information Systems and Operations Research.* 36 274-317.
29. Loulou, R. and Michaelides, E. (1979). New Greedy-Like Heuristics for the Multidimensional 0-1 Knapsack Problem. *Operations Research.* 27 1101-1114.
30. Magazine, M. and Oguz, O. (1984). A Heuristic Algorithm for the Multidimensional Zero-One Knapsack Problem. *European Journal of Operational Research.* 16 319-326.
31. Martello, S. and Toth, P. (1981). A Branch and Bound Algorithm for the Zero-One Multiple Knapsack Problem. *Discrete Applied Mathematics.* 3 275-288.
32. Merkle, D., Middendorf, M. and Schmeck, H. (2002). Ant colony optimization for resource-constrained project scheduling. *IEEE Transactions on Evolutionary Computation*. 6 333–346.
33. Nemhauser, G. and Ullmann, Z. (1969). Discrete Dynamic Programming and Capital Allocation. *Management Science.* 15 494-505.
34. Pirkul, H. (1987). A Heuristic Solution Procedure for the Multiconstraint Zero-One Knapsack Problem. *Naval Research Logistics.* 34 161-172.
35. Salkin, H.M. and Kluyver, C.A.D. (1975). The Knapsack problem: A Survey. *Naval Research Logistics*, 22(1) 127-144.
36. Senju, S. and Toyoda, Y. (1968). An Approach to Linear Programming with 0-1 Variables. *Management Science*, 15 B196-B207.
37. Shih, W. (1979). A Branch and Bound Method for the Multiconstraint Zero-One Knapsack Problem. *Journal of Operational Research Society*. 30 369-378.
38. Volgenant, A. and Zoon, J. (1990). An Improved Heuristic for Multidimensional 0-1 Knapsack Problems. *Journal of Operational Research Society.* 41 963-970.
39. Weingartner, H. and Ness, D. (1967). Methods for the Solution of the Multidimensional 0/1 Knapsack Problem. *Operations Research.* 15 83-103.
40. Zou, D., Gao, L., Li, S. and Wu, J. (2011). Solving 0–1 knapsack problem by a novel global harmony search algorithm. *Applied Soft Computing*. 11 1556-1564.