The Quadratic Multiple Knapsack Problem and Three Heuristic Approaches to It

Amanda Hiley and Bryant A. Julstrom
Department of Computer Science
St. Cloud State University
St. Cloud, MN 56301 USA
hiam0202@stcloudstate.edu, julstrom@stcloudstate.edu

ABSTRACT

The quadratic multiple knapsack problem extends the quadratic knapsack problem with K knapsacks, each with its own capacity C_k. A greedy heuristic fills the knapsacks one at a time with objects whose contributions are likely to be large relative to their weights. A hill-climber and a genetic algorithm encode candidate solutions as strings over $\{0, 1, \dots, K\}$ with length equal to the number of objects. The hill-climber's neighbor operator is also the GA's mutation. In tests on 60 problem instances, the GA performed better than the greedy heuristic on the smaller instances, but it fell behind as the numbers of objects and knapsacks grew. The hill-climber always outperformed the greedy heuristic, and on the larger instances, also the GA.

Categories and Subject Descriptors

G.2.1 [**Mathematics of Computing**]: Discrete Mathematics—*Combinatorics*; I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic Methods

General Terms

Algorithms

Keywords

Knapsack problems, quadratic multiple knapsack problem, greedy heuristic, hill-climber, genetic algorithm

1. INTRODUCTION

Knapsack problems seek to place objects in knapsacks so as to maximize the total value of the objects without overfilling a knapsack. The quadratic multiple knapsack problem extends the quadratic knapsack problem with $K > 1$ knapsacks, each with its own capacity. Here, we assume that all those capacities are the same.

Imitating a standard approach to the 0-1 knapsack problem, a greedy heuristic fills the knapsacks one at a time, always choosing the unassigned object with the highest ratio of values with other objects to its own weight that fits in the knapsack.

A stochastic hill-climber and a genetic algorithm encode candidate solutions to the problem as strings over $\{0, 1, \dots, K\}$ with length equal to the number of objects. The hill-climber's neighbor operator removes objects from each knapsack, then refills the knapsacks greedily as in the greedy heuristic. The genetic algorithm's crossover operator preserves assignments of objects to knapsacks found in both parents, then fills the offspring's knapsacks by examining the remaining objects in random order. The hill-climber's neighbor operator also serves as the GA's mutation.

The following sections of the paper describe the quadratic multiple knapsack problem; define an object's value density relative to a set of other objects and present the greedy heuristic; describe the coding of candidate solutions to the problem and operators for it; present the stochastic hill-climber and the genetic algorithm; describe a collection of problem instances; and describe and discuss the performances of the three algorithms on the instances.

2. THE PROBLEM

In the 0-1 knapsack problem (0-1 KP), we are given n objects, each with a value v_i and a weight w_i, and a knapsack with capacity C, and we seek a selection of objects for the knapsack with maximum total value but with total weight no greater than C. That is, if n binary variables x_i indicate the inclusion ($x_i = 1$) or exclusion ($x_i = 0$) of each object, we seek to maximize

$$V = \sum_{i=1}^{n} x_i v_i \tag{1}$$

while maintaining

$$W = \sum_{i=1}^{n} x_i w_i \le C. \tag{2}$$

A straightforward greedy heuristic scans the objects in increasing order of their "value densities" v_i/w_i, placing in the knapsack all the objects that fit. The heuristic returns either the total value of these objects or the value of the single most valuable objects that fits, whichever is greater.

The multiple knapsack problem (MKP) generalizes 0-1 KP by placing the objects in K knapsacks, whose capacities

C_k may be different. The goal is to maximize the total value of the objects the knapsacks contain without exceeding the capacity of any knapsack. When $K = 1$, MKP reverts to 0-1 KP.

In the quadratic knapsack problem (QKP), each object i has a value v_i and a weight w_i, as in 0-1 KP, and each pair of objects i and j has a non-negative value v_{ij} that is added to the (one) knapsack's total when it contains both object i and object j. The weight computation and capacity constraint are unchanged from 0-1 KP, to which QKP reverts when all the quadratic values v_{ij} are zero.

Combining the multiple-knapsack and quadratic generalizations of 0-1 KP yields the target of our present interest, the quadratic multiple knapsack problem (QMKP):

> Given n objects, each with a linear value v_i and a weight w_i and each pair with a quadratic value v_{ij}, and K knapsacks each with capacity C_k, the value of an assignment of objects to knapsacks is the sum of the linear values of the included objects and the quadratic values of the object pairs that fall in the same knapsack. We seek an assignment of objects to knapsacks that maximizes this sum without placing in any knapsack objects whose total weight exceeds its capacity.

That is, if $\delta_i = 1$ when object i is assigned to a knapsack, zero otherwise, and $\delta_{ij} = 1$ when objects i and j are assigned to the same knapsack, zero otherwise, QMKP seeks to maximize

$$V = \sum_{i=1}^{n} \delta_i w_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \delta_{ij} v_{ij}, \tag{3}$$

without assigning any object to more than one knapsack and without exceeding any knapsack's capacity.

Though the capacities of the knapsacks in QMKP may differ, we consider here the case in which all the capacities are the same:

$$C_1 = C_2 = \cdots = C_k.$$

This reduces the number of parameters in a problem instance.

Many researchers have described evolutionary algorithms for knapsack problems. These include Khuri, Bäck, and Heitkötter [8] and Cotta and Troya [4], for the multiple knapsack problem and Julstrom[6] for the quadratic knapsack problem, and many projects devoted to 0-1 KP. Kellerer, Pferschy, and Pisinger [7] have provided a thorough introduction to knapsack problems and their variants.

Interpretations and applications of the quadratic knapsack problem can be extended to encompass the multiple problem. As observed by, for example, Caprara, Pisinger, and Toth [3], the following graph problem gives rise to QKP. Let G be a complete, undirected graph in which each node has a value and a weight and each edge has a value, and let C be a positive bound. The value of a set of nodes is the sum of its nodes' values plus the sum of the values of the edges that join two nodes in the set. We seek a set of nodes of maximum value but with total weight no greater than C. To represent QMKP, extend this formulation with K bounds C_1, C_2, ..., C_K, and seek K mutually exclusive node sets of maximum total value, each of whose total weights does not exceed its particular bound.

Consider a manager assigned a project and a budget for salaries. She knows how productive her potential team members are likely to be, both individually and in pairs, and she seeks the most productive team within her budget. This is an instance of QKP. If the manager must assemble teams for several projects, each with its own budget, the problem becomes QMKP.

In general, QKP arises when value attaches to individual objects and pairs of objects and a constrained set of objects must be chosen with maximum value. Such situations generalize to QMKP when they require several mutually exclusive constrained sets.

3. A GREEDY HEURISTIC

The value density $vd(i, S)$ of an object i relative to a set S of other objects is the sum of the value of object i and its joint values with the objects in S divided by its weight:

$$vd(i, S) = (v_i + \sum_{j \in S} v_{ij})/w_i.$$

Note that it is possible that an object's value density relative to a set S of other objects may be zero, if the object's individual value and its quadratic values with the objects in S are all zero.

A greedy heuristic for QMKP examines relative value densities of objects to select those to assign to knapsacks. Initially, all the knapsacks are empty. For each knapsack, in order, the heuristic includes the object with highest value density relative to the remaining available objects, then fills each knapsack with objects whose value densities are high relative to the objects already there. Note that each assignment of an object to a knapsack mandates the revision of the value densities of the remaining unassigned objects relative to that knapsack's contents.

The following sketch summarizes the greedy heuristic. In it, b is an object and $vd(b, S)$ is the value density of b relative to the set S of unassigned objects.

```
all knapsacks ← $\emptyset$;
$S$ ← {all objects};
for $k$ from 1 to $K$
  $b$ ← object with maximum $vd(b, S)$;
  remove $b$ from $S$;
  add $b$ to knapsack $k$;
  for every unassigned object
    $b$ ← object with maximum $vd(b, \text{knapsack } k)$;
    if $b$ fits in knapsack $k$
      remove $b$ from $S$;
      add $b$ to knapsack $k$;
```

When all the knapsacks' capacities are the same, the order in which the knapsacks are filled does not matter. In the case of different capacities, it might well be advantageous to fill the knapsacks in an order that depends on those capacities.

4. AN EVOLUTIONARY CODING

For the quadratic multiple knapsack problem with n objects and K knapsacks, a chromosome `c[·]` is a string of length n over the alphabet $\{0, 1, 2, \ldots, K\}$ in which `c[`i`]` $= k > 0$ indicates the assignment of object i to knapsack k; `c[`i`]` $= 0$ indicates that object i is not assigned. Figure 1 illustrates the assignment of $n = 15$ objects among $K = 3$ knapsacks; note that objects 4, 7, 12, and 13 are

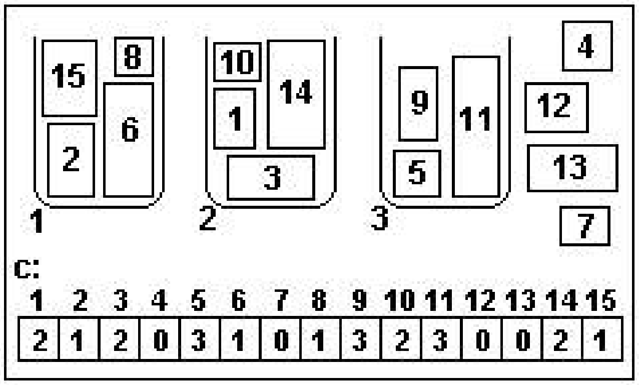

Figure 1: An assignment of $n = 15$ objects to $K = 3$ knapsacks as in the quadratic multiple knapsack problem and a chromosome that represents this assignment.

not assigned. The fitness of a chromosome, which we seek to maximize, is the sum of the values of the objects it assigns to knapsacks and of the pairs of objects it assigns to common knapsacks, as in (3).

An algorithm can enforce the knapsacks' capacity constraints by generating only representations of solutions that satisfy them, so no repair or penalty mechanism is necessary. It can generate a random valid chromosome by examining the objects in random order. For each object, it checks the knapsacks in random order and places the object in the first knapsack that can accommodate it.

A mutation operator modifies a parent chromosome to indicate the removal of some objects from some knapsacks and their replacement by random different objects. Early trials of a genetic algorithm with this operator suggested, however, that it did not implement an effective search of the problem's space. The heuristic of Section 3 inspires a greedy mutation operator that removes a number of random objects from each knapsack, then refills the knapsacks in random order. At each step, the object added to a knapsack is the unassigned object of highest value density relative to the objects already in the knapsack. This removal and refilling may increase or decrease the number of objects in each knapsack, and an object may be removed from a knapsack only to be put back. The number of objects removed from each knapsack is a parameter of the operator.

A crossover operator first copies into the offspring all the object assignments common to its two parents, then considers all the remaining objects in random order. For each object, it examines the knapsacks in random order and places the object in the first knapsack that can accommodate it, as in the generation of random chromosomes. This operator preserves assignments of objects to knapsacks that appear in both parents, and it generally creates new assignments that appear in neither parent.

5. A STOCHASTIC HILL-CLIMBER

A stochastic hill-climber for QMKP represents solutions with strings of length n, as the previous section described. It begins with a random valid solution; generates neighbors of the current solution via the heuristic mutation operator; and runs through a specified number of iterations. The number of items mutation removes from each knapsack and the number of iterations the algorithm executes are its only parameters. In the tests of Section 8 below, these values were set to two and 20,000, respectively.

6. A GENETIC ALGORITHM

A generational genetic algorithm for QMKP represents candidate solutions as strings of length n over $\{0, 1, \ldots, k\}$. It initializes its population with random valid chromosomes, and selects chromosomes to be parents in k-tournaments without replacement. It applies the crossover and heuristic mutation operators of Section 4 independently; it is elitist, preserving the best e chromosomes of the current generation unchanged into the next; and it runs through a fixed number G of generations.

In the tests that the next section describes, the GA's population contained PopSize = 100 chromosomes. The size of its selection tournaments was $k = 2$, and each tournament's winner was selected with probability 0.98, its loser with probability 0.02. The probability P[Cr] that crossover generated each offspring was 0.60, the probability of mutation therefore 0.40, and the number of items mutation removed from each knapsack was two, as in the hill-climber. The number of elite chromosomes in each generation was $e = 1$, and the GA ran through $G = 200$ generations. Note that PopSize $\times\ G = 100 \times 200 = 20,000$; the GA and the hill-climber generated equal numbers of candidate solutions.

7. SOME QMKP INSTANCES

An instance of the quadratic multiple knapsack problem consists of n objects, each with weight w_i and value v_i and each pair with value v_{ij}, and K knapsacks, each with capacity C_k. A significant feature of a QMKP instance, not to be confused with value densities of objects, is the proportion of its values v_i and v_{ij} that are non-zero. This proportion is the density of the instance, and it indicates the level of interaction among the instance's objects. If an instance's density is low, objects' contributions to the knapsacks' value tend to be independent, since each object has non-zero values with only a few other objects. If the density is high, objects' contributions depend more on the other objects in their knapsacks.

Billionnet and Soutif described exact algorithms for the quadratic (single-)knapsack problem [1] [2] and have posted on-line a collection of randomly-generated QKP instances[1]. From twenty of these, we have constructed 60 QMKP instances, with n = 100 and 200 objects and three to ten knapsacks.

In particular, Billionnet and Soutif provide twenty QKP instances with density 0.25, ten with 100 objects and ten with 200 objects. We use the first five instances from each group, setting the number of knapsacks K to three, five, and ten, to generate 30 QMKP instances. Thirty more QMKP instances are generated in the same way from the posted QKP instances with density 0.75. For each QMKP instance, the (identical) knapsack capacities are set to 80% of the sum of the instance's objects' weights divided by the number of knapsacks:

$$C_1 = C_2 = \cdots = C_K = 0.80 \frac{\sum_{i=1}^{n} w_i}{K}.$$

[1]`http://cermsem.univ-paris1.fr/soutif/QKP/QKP.html`

Table 1: Results of the trials of the greedy heuristic, the stochastic hill-climber, and the genetic algorithm on QMKP instances with density $d = 0.25$. The instances specify $N = 100$ or 200 objects that are to be assigned to $K = 3$, 5, or 10 knapsacks. The total capacity of the knapsacks is 80% of the total weight of the objects. The hill-climber and the GA were both run 40 independent times on each instance, and for each instance the table lists the best and mean values the algorithms returned as well as the standard deviation of the values and the average time for one trial. The best mean value for each instance is underlined.

				Greedy		Stochastic Hill-Climber				Genetic Algorithm			
	Instance			heuristic			Values		Mean		Values		Mean
N	K	Num	C	Value	Time	Best	Mean	StDev	time	Best	Mean	StDev	time
100	3	1	688	26554	<0.01	28144	27635	294	1.49	28665	27904	339	0.97
		2	738	24781	0.01	26915	26222	377	1.61	28059	27044	421	1.00
		3	663	24284	<0.01	25945	25193	365	1.45	26780	25991	344	0.95
		4	804	24548	<0.01	27109	26127	431	1.67	28199	27265	497	1.02
		5	723	24127	<0.01	26288	25617	297	1.58	27550	26683	397	1.00
100	5	1	413	20077	<0.01	21584	20911	289	2.00	21914	21315	316	1.20
		2	442	18119	0.01	20394	19768	322	2.11	21216	20472	326	1.25
		3	398	17742	0.01	19454	18765	255	1.99	20243	19763	296	1.17
		4	482	19104	0.01	20173	19730	279	2.30	21698	20923	291	1.31
		5	434	17694	0.01	19392	18843	266	2.12	20808	20248	259	1.25
100	10	1	206	13256	0.01	15232	14737	240	3.86	13521	12499	419	2.19
		2	221	12961	<0.01	14210	13684	243	4.11	12859	12019	299	2.31
		3	199	11624	<0.01	13334	12918	196	3.88	11790	11245	278	2.15
		4	241	12919	<0.01	14321	13867	225	4.36	13316	12593	333	2.44
		5	217	11821	<0.01	13405	12929	210	4.14	11909	11389	269	2.25
200	3	1	1381	96149	0.03	99232	98169	548	4.54	97469	95497	991	3.24
		2	1246	104867	0.03	106730	105857	469	4.38	106162	100521	3242	3.13
		3	1335	101313	0.03	103529	102475	505	4.56	101291	97157	2099	3.21
		4	1413	94053	0.02	97407	97067	831	4.55	95649	93968	812	3.29
		5	1358	99069	0.03	100827	99762	628	4.46	99458	96077	1815	3.25
200	5	1	828	68127	0.03	72277	70776	593	5.77	70731	68705	974	3.70
		2	747	75110	0.03	77551	76643	502	5.38	76297	72924	1200	3.51
		3	801	71676	0.04	75409	74497	594	5.60	74377	72924	2050	3.69
		4	848	67933	0.03	71307	69417	612	5.87	70264	67416	1138	3.83
		5	815	71424	0.02	74287	73229	465	5.79	72745	69978	1439	3.72
200	10	1	414	44443	0.02	48006	46960	609	10.16	42016	39791	982	5.77
		2	373	49214	0.02	51438	50622	404	9.09	45483	42739	1303	5.38
		3	400	47746	0.02	50717	49688	480	9.89	45698	42475	1861	5.75
		4	424	43644	0.02	47296	45751	645	10.15	41623	39446	963	5.94
		5	407	47029	0.03	50402	49431	514	9.84	46811	42399	2023	5.83

8. COMPARISONS

The greedy heuristic, the stochastic hill-climber, and the genetic algorithm were implemented in C++ and executed on a Pentium 4 processor with 1 Gbyte of memory running at 2.53 GHz under Red Hat Linux 9.0. The heuristic was run once and the two probabilistic algorithms 40 independent times on each instance. Tables 1 and 2 summarize the trials on the instances with densities $d = 0.25$ and 0.75, respectively.

An immediate observation is that the greedy heuristic is fast, requiring only a few hundredths of a second even on the larger instances. In contrast, the hill-climber and the GA require much longer; this is hardly surprising since both employ many applications of heuristic mutation and that operator imitates the greedy heuristic. The GA is consistently faster than the hill-climber. Crossover is faster than mutation, and the GA applies crossover to generate 60% of offspring chromosomes; the hill-climber applies mutation to generate every neighboring solution.

On the smaller instances with density $d = 0.25$, those with $n = 100$ objects and $K = 3$ or 5 knapsacks, the GA outperforms the hill-climber. The GA's best and mean solution values are better than those of the SHC. Both return better solutions on average than does the greedy heuristic. On the remaining (larger) instances, however, the GA falls behind the heuristic, and the hill-climber forges ahead. On average, the GA's results are almost always inferior to the heuristic's, while the hill-climber identifies solutions that are on average always better than the heuristic's, and the GA's disadvantage grows with the number of knapsacks.

On the instances with density $d = 0.75$, the hill-climber and the GA do equally well when $n = 100$ and $K = 3$ or 5; each has the best mean value on five of those ten instances, and both are always better than the greedy heuristic. On the remaining instances, again the GA falls back and the hill-climber is best. When the number of knapsacks is ten, even the GA's best result over 40 trials is inferior to the heuristic's on nine of the ten instances, though on the 200-object

Table 2: Results of the trials of the greedy heuristic, the stochastic hill-climber, and the genetic algorithm on QMKP instances with density $d = 0.75$, as in Table 1.

				Greedy		Stochastic Hill-Climber				Genetic Algorithm			
	Instance			heuristic			Values		Mean		Values		Mean
N	K	Num	C	Value	Time	Best	Mean	StDev	time	Best	Mean	StDev	time
100	3	1	669	68411	0.01	69786	69172	327	1.36	69769	68941	480	1.01
		2	714	66850	0.01	69056	68508	313	1.41	69146	68639	340	1.03
		3	686	66096	<0.01	68547	67939	361	1.46	68763	67557	832	1.02
		4	666	68261	0.01	69646	69003	427	1.46	69907	69101	861	1.04
		5	668	66693	<0.01	69480	68578	341	1.44	69410	68856	306	1.02
100	5	1	401	47628	0.01	48888	48138	286	1.97	48663	47678	749	1.26
		2	428	46411	<0.01	48686	48028	317	1.95	48990	48175	398	1.32
		3	411	45935	<0.01	47396	46970	244	1.89	47512	46623	503	1.29
		4	400	47382	0.01	49468	48864	254	1.94	49845	49194	295	1.30
		5	400	45760	<0.01	47982	47298	311	1.90	47925	47230	554	1.29
100	10	1	200	27240	<0.01	29136	28665	262	3.56	26603	25681	688	2.23
		2	214	29191	<0.01	30367	30031	187	3.86	28663	27815	391	2.43
		3	205	27110	0.01	28838	28297	238	3.76	26176	25038	562	2.31
		4	200	29259	0.01	30624	30346	199	3.89	29701	28592	333	2.40
		5	200	27993	<0.01	29375	28956	206	3.70	27130	25937	650	2.28
200	3	1	1311	263740	0.03	269447	267765	809	4.33	268919	265523	1820	3.40
		2	1414	249423	0.03	255340	253628	917	4.71	252977	249300	3409	3.47
		3	1342	264383	0.03	268682	267331	863	4.30	267731	264689	2860	3.39
		4	1565	238246	0.03	245229	243881	846	4.92	243192	237837	5179	3.64
		5	1336	273068	0.02	277221	275980	833	4.42	277762	274254	3109	3.36
200	5	1	786	176947	0.03	182374	181203	596	5.19	179525	177438	1331	3.83
		2	848	164261	0.03	172119	170505	947	5.62	168021	163917	3014	3.98
		3	805	180067	0.03	184362	182979	595	5.13	181412	178516	2361	3.83
		4	939	158161	0.03	163832	162584	745	6.13	160146	156246	3435	4.22
		5	801	186654	0.03	189756	188597	664	5.23	187333	185471	1179	3.82
200	10	1	393	108014	0.03	110238	109028	507	8.87	102002	98962	1374	5.91
		2	424	98744	0.02	102734	101595	605	9.59	92359	87400	3301	6.20
		3	402	107789	0.02	111770	110442	537	8.82	103848	100528	2362	5.95
		4	469	90556	0.02	95453	94544	532	10.71	85801	81481	2077	6.64
		5	400	110846	0.03	114260	112828	586	8.82	105078	102857	1742	5.91

instances with three and five knapsacks, the GA's results are about the same as the heuristic's. The hill-climber's average results, on the other hand, are always a few percent better than the heuristic's, and its advantage over the GA again grows with the number of knapsacks.

9. DISCUSSION

What features of the algorithms, the test instances, and their interactions might account for these results? First, the instances were generated randomly, so there are no particular patterns in their values v_i and v_{ij} nor any correlations between the values and the weights w_i. Thus objects' shared values tend to be similar, diminishing the importance of each choice of an object for a knapsack; some other object may well contribute about as much. This approximate equivalence of objects is less when an instance's density d is small; when an object interacts with fewer objects, those interactions become more important. This may account for the better performance of the GA on the 100-object instances with $d = 0.25$ than on the 100-object instances with $d = 0.75$. When an instance's density is smaller and each object's contribution to a knapsack more distinctive, crossover offers an advantage over simpler searches that diminishes as d grows.

Instances with smaller knapsacks may be more difficult for all the algorithms. When the capacities sum to 80% of an instance's total weight, most of its objects will find homes in knapsacks. If the capacities were allowed only, say, 50% of the sum of the weights, high-value assignments of objects to knapsacks might be harder to identify.

The relative performance of the GA deteriorates as the number K of knapsacks increases. When the knapsacks' capacities are equal, all numberings of the knapsacks are equivalent, so that for any representation of a solution as a string over $\{0, 1, \ldots, K\}$, there are $K! - 1$ other distinct representations of that solution. The issue is which objects are chosen and which other objects they share knapsacks with, not which knapsack each chosen object occupies. This is not a problem for the hill-climber, whose one current solution allows no ambiguity. It does, however, impede the GA, whose crossover operator can generate, from two parents that differ only in the numbering of the knapsacks, an offspring entirely unlike (and probably inferior to) both parents, so that the GA reaps benefits from crossover only when K is small. The GA might be relatively more effective on instances with more knapsacks when the knapsacks' capacities vary.

More generally, a hill-climber is likely to enjoy an advantage over an evolutionary algorithm unless the EA's recombination operator significantly advances the search of the target problem's search space. A hill-climber conserves every improvement that its solution encounters; an EA may generate many solutions that do not advance the search.

10. CONCLUSION

In the quadratic multiple knapsack problem, n objects have individual values v_i and weights w_i, pairs of objects have quadratic values v_{ij}, and K knapsacks have capacities C_k. The value of an assignment of objects to knapsacks is the sum of the values of the assigned objects and of the quadratic values of pairs of objects assigned to the same knapsack. The goal is an assignment of objects to knapsacks that maximizes this sum without placing in any knapsack objects whose total weight exceeds the knapsack's capacity. Here, all the knapsacks' capacities are equal.

A greedy heuristic fills the knapsacks one at a time. Each knapsack's first object has the largest value density, among objects that fit, relative to the objects not yet assigned to a knapsack; each remaining object has the largest value density, among objects that fit, relative to the objects already in that knapsack. A stochastic hill-climber and a genetic algorithm encode candidate assignments of objects to knapsacks as strings of length n over $\{0, 1, \ldots, K\}$. The hill-climber's neighbor operator, which is also the GA's mutation, removes some objects from each knapsack, then refills the knapsacks as in the greedy heuristic. The GA's crossover preserves assignments of objects to knapsacks that are common to its two parents, then fills the offspring's knapsacks randomly.

In comparisons on 60 randomly-generated instances of QMKP, both the hill-climber and the GA returned better solutions on average than did the greedy heuristic, though at much greater computational expense. On larger instances, the GA fell behind both the heuristic and the hill-climber, but the latter continued to identify better solutions than did the heuristic. These results are likely due to (1) the random nature of the instances; (2) the redundancy of the coding, particularly when K is large; and (3) the inherent advantage of hill-climbing, which conserves all good news.

This study can be extended and refined in many ways. In addition to the conjectures in the previous section, the hill-climber can be extended with a tabu list or by choosing each candidate solution from a selection of neighbors. Both the hill-climber and the GA might be more effective with a less-redundant coding, and both could be seeded with the greedy heuristic's solution. All the algorithms should be tested on instances designed to mislead the greedy heuristic and instances whose knapsack capacities differ.

11. REFERENCES

[1] Alain Billionnet and Éric Soutif. An exact method based on Lagrangian decomposition for the 0-1 quadratic knapsack problem. *European Journal of Operational Research*, 157(3):565–575, 2004.

[2] Alain Billionnet and Éric Soutif. Using a mixed integer programming tool for solving the 0-1 quadratic knapsack problem. *INFORMS Journal on Computing*, 16(2):188–197, 2004.

[3] Alberto Caprara, David Pisinger, and Paolo Toth. Exact solution of the quadratic knapsack problem. *INFORMS Journal on Computing*, 11:125–137, 1999.

[4] Carlos Cotta and José Ma Troya. A hybrid genetic algorithm for the 0-1 multiple knapsack problem. In G. D. Smith, N. C. Steele, and R. F. Albrecht, editors, *Artificial Neural Networks and Genetic Algorithms 3*, pages 250–254. Springer-Verlag, Berlin, 1998.

[5] Ed Deaton, Dave Oppenheim, Joseph Urban, and Hal Berghel, editors. *Applied Computing 1994: Proceedings of the 1994 ACM Symposium on Applied Computing*, New York, 1994. ACM Press.

[6] Bryant A. Julstrom. Greedy, genetic, and greedy genetic algorithms for the quadratic knapsack problem. In Hans-Georg Beyer et al., editors, *GECCO 2005: Genetic and Evolutionary Computation Conference*, pages 607–614, New York, 2005. ACM Press.

[7] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*. Springer, Berlin, 2004.

[8] Sami Khuri, Thomas Bäck, and Jörg Heitkötter. The zero/one multiple knapsack problem and genetic algorithms. In Deaton et al. [5], pages 188–193.