EasyChair Preprint

№ 437

An Iterated Semi-Greedy Algorithm for the 0-1 Quadratic Knapsack Problem

Leticia Leonor Pinto Alva and Alexander J. Benavides

August 17, 2018

An Iterated Semi-Greedy Algorithm for the 0-1 Quadratic Knapsack Problem

Leticia L. Pinto Alva
Universidad Católica San Pablo, Arequipa, Perú
leticia.pinto@ucsp.edu.pe

Alexander J. Benavides
Universidad Católica San Pablo, Arequipa, Perú.
Universidad Nacional de San Agustín de Arequipa, Perú.
ajbenavides@ucsp.edu.pe, ajbenavides@ucsp.edu.pe

May 2018

Abstract

This paper presents a new Iterated Semi-Greedy Algorithm (ISGA) for the 0-1 Quadratic Knapsack Problem. The proposed ISGA is easier to implement, runs faster, and produces comparable results than state-of-the-art methods. Computational evaluation is performed over a benchmark with large instances of 1000 and 2000 objects.

Keywords: 0-1 Quadratic Knapsack Problem; Metaheuristics; Greedy algorithms.

1 Introduction

Problems in the operations research area consist in finding an optimal configuration (subset or permutation) according to an optimization criteria. When the problem is considered NP, the size of the search space grows exponentially, and neither enumerative nor exact methods are capable of solving large instances. Under those circumstances, the use of approximate heuristic methods is necesary to produce good quality solutions within a reasonable computational time.

The Quadratic Knapsack Problem (QKP) is a combinatorial optimization problem, and it belongs to the NP problems class. The best metaheuristic method for the QKP in the literature is a GRASP+T proposed by Yang, Wang, and Chu [4]. In this paper we present a simple Iterated Greedy Algorithm that is easier to implement, and reaches similar results. The rest of the paper is organized as follows: The next section defines the QKP. Section 3 reviews the best methods in the literature. Section 4 describes the proposed Iterated Semi-Greedy Algorithm. Section 5 shows computational results. Section 6 presents our concluding remarks and hints our next research steps.

2 The Quadratic Knapsack Problem

The Quadratic Knapsack Problem (QKP) is a nonlinear combinatorial optimization problem with many applications in multiple discipline areas. Given a capacity-constrained knapsack and a set of candidate objects, where each object has a positive weight and produces a profit if selected, we must select a subset of objects to fill the knapsack, trying to maximize the overall profit, and

without surpassing the knapsack capacity constraint. In the QKP, there is a pairwise profit besides the profit produced by each independent object.

The 0-1 Quadratic Knapsack Problem was introduced by Gallo, Hammer, and Simeone [2]. Let us assume that the knapsack's capacity is c and there are n candidate objects, that w_i and p_{ii} respectively are the weight and the profit of object $i \in [n]$, and that p_{ij} is the pairwise profit of two objects $i \in [n]$ and $j \in [n]$ ($i \neq j$). Let the binary string $X = (x_1, \ldots, x_n)$ represent a solution, where the binary variable x_i determines whether the object i is in the knapsack or not. Then, a mathematical definition for the 0-1 QKP is

$$\textbf{maximize} \qquad f(X) = \sum_{i \in [n]} \sum_{j \in [n]} p_{ij} x_i x_j, \tag{1}$$

$$\textbf{subject to} \qquad \sum_{i \in [n]} w_i x_i \leq c, \tag{2}$$

$$x_i \in \{0, 1\}, \qquad \forall i \in [n], \tag{3}$$

where Equation (1) is the profit function to be maximized, Equation (2) ensures the weight capacity limit of the knapsack.

3 Literature review

Julstrom [3] proposes 4 methods for the 0-1 QKP: two greedy heuristics and two genetic algorithms (GA). The greedy heuristics build solutions by inserting objects in the knapsack in a non-increasing order of their value densities. The GAs encode efficiently selections of objects as binary strings, and produce only strings with a total weight that do not surpasses the knapsack's capacity. One GA does not use information about the values associated with the objects in the knapsack, the other embeds greedy techniques to produce better results. These methods were tested with small instances (with 100 and 200 objects).

Yang, Wang, and Chu [4] propose a Greedy Randomized Adaptive Search Procedure (GRASP+T) followed by a tabu search for the 0-1 QKP. Their algorithm iterates over three main steps: a construction phase, a local search, and updating the best so far. The construction phase builds up an initial solution by iteratively inserting the most frequent object among the best fitted objects. The local search explores feasible solutions using shift and swap neighbourhoods. At each iteration, it updates the best solution so far if the local minimum is better. Finally, a simple tabu search algorithm is performed over the best solution so far. It searches the shift and swap neighbourhoods, accepting worse solutions while forbidding the elimination of recently included objects. The prohibitions are ignored if a move leads to a better solution.

Their construction phase uses two priority rules to select an object $j \in R(S)$ to be inserted into the knapsack. Let set S represent the objects in the knapsack (object i is in the knapsack if $x_i = 1$). The first priority rule is a non-increasing order of the greedy function

$$f_2(S, j) = \frac{obj(S) + \sum_{i \in S} p_{ij} + p_{jj}}{sw(S) + w_j}, \qquad \forall j \in R(S), \tag{4}$$

where set $R(S)$ has only objects that fit into the knapsack (an object j fits into the knapsack if $x_j = 0$ and it does not exceed the capacity $sw(S) + w_j \leq c$), $sw(S)$ is the total weight of the objects in S, w_j is the weight of object j, $obj(S)$ is the objective function of the current solution S, and p_{ii} and p_{ij} are the aforementioned profit values. Their construction phase picks a random number of objects with that priority rule, and uses the frecuency of the objects in the knapsack as a second priority rule to select most frequent object among the best fitted objects.

Other methods for the 0-1 QKP include a global harmony search algorithm [1] and a dynamic programming [5]. Not all of their results are directly comparable because they used different instances. We consider that the best method is the GRASP+T of Yang, Wang, and Chu [4], and we compare our computational results to theirs in Section 5.

4 An Iterated Semi-Greedy Algorithm for the 0-1 QKP

An Iterated Greedy Algorithm (IGA) is a simple but effective method that can be used to solve the 0-1 QKP. The initial solution is produced with a greedy constructive heuristic (described in Section 4.1) and then it is improved by a local search (described in Section 4.3). When an initial solution is ready, the **IGA** iterates over two main steps: A greedy perturbation phase, and an optional improvement phase (same local search). This paper proposes an Iterated Semi-Greedy Algorithm (ISGA) for the 0-1 QKP, that performs a semi-greedy perturbation phase (described in Section 4.2). At the end of each iteration, the best solution so far is replaced with the current solution if there is an improvement; otherwise, the current solution rolls back to the best solution so far. The stop criterion for the ISGA is $4n$ iterations, where n is the number of objects.

4.1 Greedy Constructive Heuristic

The greedy constructive heuristic generates an initial solution by inserting one element at a time into a partial solution. The element which brings the maximum ratio of benefit over weight is inserted at each iteration, until no more elements fit into the knapsack. The ratio of benefit over weight is evaluated with the priority rule f_2 of Yang, Wang, and Chu [4] defined in Equation (4). When a solution is complete, we perform a local search.

4.2 Semi-Greedy Perturbation

The semi-greedy perturbation phase removes d random elements from the knapsack. Then, it iteratively reinserts into the knapsack a random element among those with best insertion benefit over the profit function. This is repeated until no more elements fit into the knapsack.

The insertion benefit that an object j produces when it is inserted into the partial solution S is

$$\Delta f(S, j) = \sum_{i \in [n]} p_{ij} x_i + p_{jj}, \qquad \forall j \in R(S). \tag{5}$$

Then the semi-greedy perturbation randomly inserts an object j that has an insertion benefit $\Delta f(S, j) \geq p * \max_\Delta$, where $\max_\Delta = \max_{j \in R(S)} \Delta f(S, j)$ is the maximum insertion benefit, and parameter p cuts a percentage from the value of $\max_\Delta$. Consequently, a value of $p = 100\%$ is equivalent to a greedy perturbation, and a value of $p = 0\%$ is equivalent to a random perturbation.

4.3 Local Search

This improvement method explores the swap neighbourhood; i.e. it swaps an element in the knapsack with one out of it, respecting the capacity limit, and looking for an improvement in the profit function.

5 Computational Results

5.1 Experimental Methodology

The ISGA was implemented in C++11, compiled with the GNU C++ compiler version 4.8.5 (Red Hat 4.8.5-4) with optimization level 3, and run on a PC with an Intel(R) Xeon(R) CPU E5-2680 v4 processor running at 2.40 GHz, and with 64 GB of main memory, using only one core for each experiment.

Table 1: ARD (in thousandths) for the calibration of parameters p and d.

p	d			
(%)	1	2	3	4
50	2.561	2.120	3.350	4.284
55	1.621	1.918	3.530	4.509
60	2.624	1.834	3.260	4.207
65	2.565	2.455	3.442	4.339
70	2.811	2.285	3.437	4.448
75	2.066	2.534	3.379	4.418
80	2.709	2.732	3.342	4.460
85	2.200	2.645	3.600	4.588
90	1.949	2.305	3.150	4.517
95	2.261	2.187	2.789	4.638
100	4.586	4.357	4.824	5.159

Those experiments were developed in the Center for High Computational Performance of the Peruvian Amazon from "Instituto de Investigaciones de la Amazonía Peruana" (**IIAP**). More information : `http://iiap.org.pe/manati`.

Section 5.3 compares our results to those of Yang, Wang, and Chu [4]. We have tested the ISGA on the same instances tested by Yang, Wang, and Chu [4] (with 1000 and 2000 objects). The ISGA tests were replicated 100 times on each instance, to match the experiments of Yang, Wang, and Chu [4].

We present the quality of the results as the relative deviation RD $= (f^* - f)/f^*$ from the best values f^* reported by Yang, Wang, and Chu [4], and as the average relative deviation (ARD) for groups of instances and test replications.

5.2 Calibration of Parameters

The ISGA has two parameters: parameter d determines the number of elements to be deleted in the perturbation phase, and parameter p determines which percentage of $\max_\Delta$ is used as limit to randomly select the next object for the knapsack. We test parameter $d \in \{1, 2, 3, 4\}$ and parameter $p \in \{50\%, 55\%, \ldots, 100\%\}$. We randomly selected one instace from each group with 1000 objects for the calibration. Instances 1000_25_6, 1000_50_4, 1000_75_7, and 1000_100_3 were selected. Table 1 shows the ARD values for each parameter level. There is not a clear tendency for any isolated parameter. The best ARD of 1.621 (highlighted in gray) is achieved with $p = 55\%$ and $d = 1$, thus we set these values for the rest of the experiments.

5.3 Experimental Results

ISGA and GRASP+T show similar results, with an overall average relative deviation of 0.354 thousandths for instances with 1000 objects, and 0.071 for instances with 2000 objects. Tables 2 and 3 show the best solution found for each instance by GRASP+T and ISGA in 100 replications, and the corresponding minimum and average relative deviations, dividing the instances with 1000 and 2000 objects. Negative RD values indicate that ISGA finds better results than GRASP+T. ISGA finds a better result in all replications for instance 1000_100_6, and in 87% of the replications for instance 2000_100_2. ISGA finds the same results than GRASP+T for half of the instances (22 with 1000 objects and 19 with 2000 objects). Tables 2 and 3 also show the average runtime for GRASP+T and ISGA. GRASP+T has an overall average runtime of 288 seconds, and ISGA of 94

Table 2: Best profit, RD (in thousandths), and average runtime for each instance with 1000 objects.

	Best Solutions		ISGA's RD		Avg. Time	
Instance	GRASP+T	ISGA	Minimum	Average	GRASP+T	ISGA
1000_25_1	6172407	6172239	0.027	0.056	29.41	8.49
1000_25_2	229941	229941	0.000	0.359	32.98	18.74
1000_25_3	172418	172418	0.000	0.219	21.66	13.79
1000_25_4	367426	367426	0.000	0.000	26.11	13.38
1000_25_5	4885611	4885158	0.093	0.117	37.69	23.45
1000_25_6	15689	15689	0.000	4.859	8.18	15.83
1000_25_7	4945810	4944857	0.193	0.226	36.51	23.12
1000_25_8	1710198	1710198	0.000	0.098	71.21	19.07
1000_25_9	496315	496315	0.000	0.000	30.03	13.96
1000_25_10	1173792	1173792	0.000	0.148	58.93	20.30
1000_50_1	5663590	5663590	0.000	0.034	50.74	15.93
1000_50_2	180831	180824	0.039	0.039	1.44	16.60
1000_50_3	11384283	11384283	0.000	0.022	31.86	11.26
1000_50_4	322226	321593	1.964	2.277	22.06	14.46
1000_50_5	9984247	9984155	0.009	0.055	40.83	13.58
1000_50_6	4106261	4106261	0.000	0.068	58.08	16.32
1000_50_7	10498370	10498286	0.008	0.038	33.43	11.68
1000_50_8	4981146	4980460	0.138	0.223	116.29	22.23
1000_50_9	1727861	1727756	0.061	0.105	52.77	12.30
1000_50_10	2340724	2340579	0.062	0.147	95.28	14.72
1000_75_1	11570056	11570018	0.003	0.047	64.00	14.59
1000_75_2	1901389	1901389	0.000	0.143	32.47	11.16
1000_75_3	2096485	2092253	2.019	2.147	39.86	13.78
1000_75_4	7305321	7305315	0.001	0.076	55.09	13.47
1000_75_5	13970240	13969984	0.018	0.043	37.39	15.22
1000_75_6	12288738	12288738	0.000	0.029	33.44	14.36
1000_75_7	1095837	1094190	1.503	1.630	23.16	13.06
1000_75_8	5575813	5575813	0.000	0.097	68.47	15.59
1000_75_9	695774	695774	0.000	0.205	22.68	15.89
1000_75_10	2507677	2507677	0.000	0.128	47.32	11.92
1000_100_1	6243494	6243494	0.000	0.049	72.01	14.09
1000_100_2	4854086	4854086	0.000	0.065	84.84	15.09
1000_100_3	3172022	3172022	0.000	0.112	47.06	13.26
1000_100_4	754727	754727	0.000	0.138	23.63	12.37
1000_100_5	18646620	18646540	0.004	0.018	39.15	11.65
1000_100_6	16018298	16020049	-0.109	-0.081	41.58	13.46
1000_100_7	12936205	12936205	0.000	0.020	44.50	10.88
1000_100_8	6927738	6927671	0.010	0.059	96.05	13.64
1000_100_9	3874959	3874959	0.000	0.032	52.28	11.97
1000_100_10	1334494	1334494	0.000	0.101	23.63	12.06

Table 3: Best profit, RD (in thousandths), and average runtime for each instance with 2000 objects.

	Best Solutions		ISGA's RD		Avg. Time	
Instance	GRASP+T	ISGA	Minimum	Average	GRASP+T	ISGA
2000_25_1	5268188	5268117	0.013	0.054	516.57	272.57
2000_25_2	13294030	13292445	0.119	0.147	330.73	229.96
2000_25_3	5500433	5500213	0.040	0.131	800.13	276.27
2000_25_4	14625118	14625118	0.000	0.016	346.89	213.25
2000_25_5	5975751	5975058	0.116	0.182	738.33	302.87
2000_25_6	4491691	4491691	0.000	0.000	474.60	188.25
2000_25_7	6388756	6388756	0.000	0.023	558.21	215.65
2000_25_8	11769873	11768743	0.096	0.124	446.95	248.12
2000_25_9	10960328	10958714	0.147	0.175	449.81	248.60
2000_25_10	139236	139236	0.000	0.000	109.79	129.84
2000_50_1	7070736	7067526	0.454	0.608	474.32	192.28
2000_50_2	12587545	12587545	0.000	0.054	534.87	209.61
2000_50_3	27268336	27268336	0.000	0.000	308.88	122.34
2000_50_4	17754434	17754320	0.006	0.061	782.66	212.22
2000_50_5	16805490	16805288	0.012	0.066	1490.22	195.57
2000_50_6	23076155	23076155	0.000	0.025	460.09	165.85
2000_50_7	28759759	28757686	0.072	0.099	714.18	149.07
2000_50_8	1580242	1580242	0.000	0.331	165.18	124.23
2000_50_9	26523791	26523476	0.012	0.040	342.12	151.20
2000_50_10	24747047	24747047	0.000	0.000	408.39	134.33
2000_75_1	25121998	25121998	0.000	0.023	807.05	152.68
2000_75_2	12664670	12664670	0.000	0.036	510.05	175.47
2000_75_3	43943994	43943599	0.009	0.017	276.39	132.31
2000_75_4	37496613	37496600	0.000	0.023	354.13	114.39
2000_75_5	24834948	24834904	0.002	0.032	684.33	184.03
2000_75_6	45137758	45137758	0.000	0.000	306.47	101.73
2000_75_7	25502608	25502608	0.000	0.022	490.14	139.69
2000_75_8	10067892	10067892	0.000	0.032	344.83	124.24
2000_75_9	14171994	14171584	0.029	0.088	532.06	186.63
2000_75_10	7815755	7815755	0.000	0.072	325.22	135.00
2000_100_1	37929909	37929909	0.000	0.018	435.71	123.51
2000_100_2	33647322	33648033	-0.021	-0.010	791.51	194.28
2000_100_3	29952019	29951979	0.001	0.019	1489.29	145.66
2000_100_4	26949268	26949020	0.009	0.045	710.79	156.64
2000_100_5	22041715	22041691	0.001	0.020	752.02	188.74
2000_100_6	18868887	18868860	0.001	0.043	548.19	142.13
2000_100_7	15850597	15850594	0.000	0.012	578.18	145.54
2000_100_8	13628967	13628967	0.000	0.071	374.07	136.28
2000_100_9	8394562	8394562	0.000	0.063	304.31	114.95
2000_100_10	4923559	4923559	0.000	0.088	200.05	131.32

seconds. Taking into account that our platform is a factor of 1.39 faster, our algorithm runs in a corresponding time or less.

6 Conclusions

This paper presented a new Iterated Semi-Greedy Algorithm (ISGA) for the 0-1 QKP. ISGA is easier to implement, runs faster, and produces similar results than the state-of-the-art GRASP+T of Yang, Wang, and Chu [4].

Our next research includes two aspects: to study the inclusion of the simple tabu search proposed by Yang, Wang, and Chu [4] to replace the local search, and to study priority rules for the elimination of objects in our perturbation phase. We believe those changes might improve our current results.

References

[1] Franklin Djeumou Fomeni and Adam N Letchford. A dynamic programming heuristic for the quadratic knapsack problem. *INFORMS Journal on Computing*, 26(1):173–182, 2013.

[2] Giorgio Gallo, Peter L Hammer, and Bruno Simeone. Quadratic knapsack problems. In *Combinatorial optimization*, pages 132–149. Springer, 1980.

[3] Bryant A Julstrom. Greedy, genetic, and greedy genetic algorithms for the quadratic knapsack problem. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 607–614. ACM, 2005.

[4] Zhen Yang, Guoqing Wang, and Feng Chu. An effective grasp and tabu search for the 0–1 quadratic knapsack problem. *Computers & Operations Research*, 40(5):1176–1185, 2013.

[5] Dexuan Zou, Liqun Gao, Steven Li, and Jianhua Wu. Solving 0–1 knapsack problem by a novel global harmony search algorithm. *Applied Soft Computing*, 11(2):1556–1564, 2011.