

Determining when Gurobi can Optimally Solve the Quadratic Knapsack Problem with Conflict Graphs: Implications for OR Practitioners

AMBER KULP¹, YUN LU², MYUNG SOON SONG², FRANCIS J. VASKO²

¹Department of Computer Science & Information Technology,
Kutztown University,
Kutztown, PA, 19530,
USA

²Department of Mathematics,
Kutztown University,
Kutztown, PA, 19530,
USA

Abstract: - The Quadratic Knapsack Problem with Conflict Graphs (QKPCG) was introduced in a 2017 paper along with 45 QKPCG test instances. Since then, none of the solution methods presented in the literature have been designed to generate proven optimums or even any measure of how close the solutions are to the optimums. In this paper, the original 45 QKPCG test instances are extended to a more comprehensive set of 810 test instances (765 new instances). The reason for expanding the test instances is that the original 45 QKPCG instances were very limited in scope. This more comprehensive set of test instances provides the opportunity to determine, when using general-purpose integer programming software (Gurobi), how solution difficulty changes when the conflict graph density changes, as well as the density of the quadratic profit terms (p_{ij} s). Since Gurobi is widely used in industry to solve optimization problems, these results can provide practical knowledge to operations research (OR) practitioners. Additionally, the benefit of using solutions generated by a hybrid search method (HSM) available in the literature as input to Gurobi when the p_{ij} densities are 75% or greater is demonstrated. However, when the p_{ij} densities are less than 75%, the use of HSM solutions as input to Gurobi is shown to be of no benefit. Over the 810 test instances, Gurobi determined proven optimums for over 93% of them (99% when the density of the quadratic profit terms is less than 75%). Finally, although no new theory or exact algorithm is presented, this is the first time a QKPCG solution procedure that strives to obtain proven optimums is discussed in the literature. The simplicity and ease of implementation of this approach make it very attractive for use by OR practitioners.

Key-Words: - Quadratic knapsack problem; conflict graphs; test instances; Gurobi; hybrid search method (HSM); Operations Research Practice; exact solution methods; approximate solution methods.

Received: April 13, 2026. Revised: May 6, 2026. Accepted: June 28, 2026. Published: October 21, 2026.

1 Introduction

In [1] the quadratic knapsack problem (QKP) is introduced. The quadratic knapsack problem (QKP) is an extension of the basic knapsack problem (KP) in which, in addition to a profit for inserting an item into the knapsack without exceeding the knapsack capacity, if certain pairs of items are inserted into the knapsack, then an additional profit is realized. A comprehensive survey of the QKP was provided in [2]. A more current paper [3] gave an overview of recent advances dealing with the QKP. Recently published papers dealing with exact solution methods for the QKP include a specialized cut-and-

branch algorithm [4], and a lifted-space dynamic programming algorithm [5].

Traditionally, if a QKP was solved and two items in the solution were in conflict (could not both be in the solution), then this conflict was resolved after the QKP solution had been generated. In order to handle these conflicts as part of the solution process, the quadratic knapsack problem with conflict graphs (QKPCG) was introduced in a 2017 paper [6]. In [6] the QKPCG is used as a model for designing and optimizing environmental economic policies in a protective zone. The conflict graph (items are vertices, and edges connect items in

conflict) of a QKPCG is used to model that pairs of items connected by an edge in the conflict graph cannot both be inserted into the knapsack. The QKPCG extends the basic KP by considering both quadratic profits and conflicts between pairs of items.

More precisely, in a QKPCG, there is a knapsack with capacity c , a set of n items with weight w_i and profit p_i for item i and a set E of pairs of incompatible items. Additionally, there is a nonnegative profit p_{ij} ($i < j$) associated with each pair of compatible items. The p_{ij} density is the percentage of p_{ij} values that are positive. The purpose of the QKPCG is to maximize the total profit of compatible items placed in the knapsack without exceeding capacity.

Mathematically, the QKPCG can be described as follows:

$$\max \sum_{i \in I} p_i x_i + \sum_{i \in I} \sum_{j \in I, j > i} p_{ij} x_i x_j \quad (1)$$

$$\text{s.t.} \quad \sum_{i \in I} w_i x_i \leq c \quad (2)$$

$$x_i + x_j \leq 1, \quad \forall (i, j) \in E \quad (3)$$

$$x_i \in \{0, 1\}, \quad \forall i \in I \quad (4)$$

where x_i is equal to 1 if item i is inserted in the knapsack and 0 otherwise. The objective function (1) is composed of both linear terms and quadratic terms. Constraint (2) ensures that the knapsack capacity is not exceeded. Constraint set (3) ensures that all items inserted into the knapsack are compatible, and constraint set (4) ensures that the variables are binary.

This formulation can be used to model the problem of trying to establish an optimal economic development strategy in an environmental protection zone. Specifically, suppose that a development plan is composed of n components. Let $x_i = 1$ mean that the i th component is applied, $x_i = 0$ otherwise. Each component of the plan represents a natural resource that can be used for an investment. The objective function measures the total reward based on the investments' performance. The capacity constraint ensures that the natural resources used do not exceed the fixed ecological boundary. The objective function quadratic term denotes the additional profit whenever both investments i and j are in the solution. Finally, incompatibility constraints arise when two different investments cannot share the same natural resource. We suggest the reader consult [6] for more details on this motivation of the QKPCG formulation.

Since the introduction of the QKPCG in [6] in 2017, all papers in the literature have dealt with only

solving the 45 QKPCG test instances that were also introduced in [6] and have focused only on approximate solution procedures that do not provide any guarantees on solution quality.

In this paper, we will first extend the 45 QKPCG test instances to a more comprehensive set of 810 test instances (765 new instances). The purpose of this expansion is to systematically determine, when using general-purpose integer programming software (Gurobi), how solution difficulty changes when the conflict graph density changes as well as the density of the quadratic profit terms. To the best of our knowledge, this form of structured sensitivity analysis has not appeared in the QKPCG literature. Then we will demonstrate when there is a benefit from inputting solutions generated by a hybrid search method (HSM) from the literature to the general-purpose integer programming software Gurobi to solve QKPCG instances. Since HSM is available in literature and Gurobi is widely used in industry to solve optimization problems, these results can provide practical knowledge to operations research (OR) practitioners.

This paper makes the following contributions: (1) it introduces a systematically expanded benchmark set of 810 QKPCG instances derived from the original 45 instances; (2) it provides the first empirical study analyzing how conflict graph density and quadratic profit density (p_{ij}) impact the ability of a general-purpose solver to obtain proven optimal solutions; (3) it evaluates the effectiveness of warm-starting Gurobi using HSM solutions; and (4) it provides practical insights into when QKPCG instances are efficiently solvable using standard optimization software.

We emphasize that, while elements of the computational strategy (e.g., use of default Gurobi settings) are consistent with the prior work [16], the present study is novel in its focus on QKPCG structure, expanded benchmark design, and solvability analysis.

In the next section, we will review the literature related specifically to the QKPCG. Then we will focus on the test instances. First, we will review the 45 QKPCG instances introduced in [6], and then we will extend these instances to a set of 810 instances which include the 45 from [6]. This will be followed by a discussion of when it is beneficial to employ both the general-purpose integer programming Gurobi software package and the best-performing approximate solution procedure from the literature. The goal of this simple methodology is to try to generate proven optimums for QKPCG instances—something that has not been done

previously but has important industrial implications for OR practitioners. Finally, we will provide some observations and possible future work.

2 Literature Review

In this section, we will focus exclusively on reviewing the QKPCG literature. To our knowledge, there have been only four papers that have been published through April 2026 that deal with the QKPCG: the 2017 paper [6], a 2020 paper [7], a 2021 paper [8], and a 2024 paper [9]. For a comprehensive overview of the many extensions of the basic KP that involve a single knapsack, the reader is directed to [10], and for extensions that involve multiple knapsacks, the reader is directed to [3].

In addition to providing the mathematical formulation of the QKPCG and defining 45 QKPCG test instances, [6] provided a construction heuristic followed by a neighborhood search that was called NSBM. For their 45 test instances, NSBM gave better results compared to executing the software package GLPK for one hour. In [7] a population-based particle swarm optimization (PSO) heuristic followed by a neighborhood search is used. This algorithm was referred to as PBSA. Again, using only the [6] 45 test problems, PBSA gave somewhat better results than NSBM. The third paper presenting a solution approach for the QKPCG was [8] and its solution approach combined an intensification search with a descent method and a diversification strategy for enlarging the search space. This method was called MDM and was compared to NSBM, with MDM slightly outperforming NSBM on the same 45 test instances. The final published paper that we are aware of is [9] that presents a hybrid search method referred to as HSM. This hybrid method is based on a memetic framework and possesses the following characteristics: (1) it applies a dedicated randomized uniform-based crossover operator (RUX), which strives to preserve common items shared by parent solutions, (2) it efficiently explores using a tabu search procedure the search space around each newly generated solution by the crossover operator, and (3) a fast incremental evaluation technique is used to quickly evaluate each candidate move. On the same 45 test instances, HSM outperformed NSBM, MDM, and PBSA.

It is important to note that the four solution approaches discussed above (NSBM, PBSA, MDM, and HSM) are stochastic in nature, and each is typically executed 10 times, and the best solution is reported as the final answer. None of these solution

methods guarantees optimal solutions or even bounds on the solutions generated. In fact, in [9] it is stated that “HSM cannot ensure the optimality of the obtained solutions...”. Furthermore, as will be discussed in the next section, it is important to note that all published solution methods for the QKPCG test their approaches on only the 45 QKPCG test instances introduced in [6]. The purpose of this paper is twofold: (1) to define a comprehensive set of 810 QKPCG test instances, of which the 45 defined in [6] is a subset, and (2) to determine when providing an initial solution (HSM) is beneficial in generating guaranteed optimums for these 810 test instances using the general-purpose integer programming software Gurobi. Hence, this paper will be the first to focus on obtaining proven optimums for the QKPCG.

Beyond QKPCG-specific studies, the broader quadratic knapsack literature includes exact approaches such as branch-and-bound, cutting-plane methods, and dynamic programming-based techniques [2], [4], [5]. However, these methods have not been extended to incorporate conflict graph constraints in a way that produces proven optimal solutions for QKPCG benchmark instances. This gap motivates the present study.

In the next section, after discussing the characteristics of the 45 [6] test instances, we will extend these to a more comprehensive set of test instances.

3 Quadratic Knapsack Problem with Conflict Graphs (QKPCG) Test Instances

3.1 The Original 45 QKPCG Test Instances ([6])

When in [6] the the QKPCG and a heuristic solution procedure to solve the QKPCG were defined, 45 QKPCG test instances that they used to test the performance of their solution procedure were also defined. These 45 test instances are the only instances that have been used to test QKPCG solution procedures [7], [8], [9] published in the literature as of April 2026. The characteristics of these 45 QKPCG instances are given in Table 1.

All 45 instances have a knapsack capacity of 1000. There are 9 classes with 5 instances in each class. There are 15 instances with 100 items, 15 instances with 150 items, and 15 instances with 200 items. As can be seen from Table 1, the conflict graph densities are 2%, 4%, and 8%.

What is not mentioned in [6], but can easily be deduced by examining the 45 data sets, is that all 45 instances have p_{ij} densities of 100%, where p_{ij} density refers to the percentage of positive p_{ij} values ([4], [5], [11], [12], [13]). Even items that are incompatible and cannot both be inserted into the knapsack have positive p_{ij} values in [6] data sets. For these 45 data sets, we set $p_{ij} = 0$ if items i and j are incompatible. Furthermore, it is very common ([4], [5], [11], [12], [13]) in the quadratic knapsack problem literature to consider the impact of having p_{ij} densities less than 100%. In the literature, the impact of p_{ij} densities has not been considered when solving QKPCG instances. This paper will be the first to consider the impact of p_{ij} densities when solving QKPCG instances. In particular, this paper will focus on the impact p_{ij} density has on the ability of Gurobi to obtain guaranteed optimums for QKPCG instances.

In the next subsection, these 45 instances will be increased to 810 instances by varying the p_{ij} densities and the conflict graph densities.

Table 1. Description of the original 45 benchmark QKPCG instances (5 instances per class and knapsack capacity 1000 for all instances).

Class	n	Conflict graph density	p_{ij} density
1qkpcg	100	2%	100%
2qkpcg	100	4%	100%
3qkpcg	100	8%	100%
4qkpcg	150	2%	100%
5qkpcg	150	4%	100%
6qkpcg	150	8%	100%
7qkpcg	200	2%	100%
8qkpcg	200	4%	100%
9qkpcg	200	8%	100%

Source: [6]

3.2 The 765 New QKPCG Instances

Our strategy for increasing the number of QKPCG instances from [6] was to use these instances but to modify the p_{ij} densities and the conflict graph densities. Increasing the number and variety of QKPCG test instances is consistent with prior work on expanding benchmark sets [14], [15]. Hopefully, by solving these new QKPCG instances, we will get some indication of how solution difficulty changes when the conflict graph density changes as well as the density of the quadratic profit terms. Specifically, p_{ij} values were randomly selected from the original 45 QKPCG instances. To improve reproducibility, the generation process can be described more precisely as follows: for each instance, p_{ij} values are randomly set to zero until the

desired density level (90%, 75%, 50%, 25%, or 0%) is achieved, while preserving the remaining structure. We included 0% case, which reduces the problem to a knapsack problem with a conflict graph, to provide a baseline comparison. Similarly, conflict graph densities are modified by randomly removing edges to halve the density to 1%, 2%, and 4%, or by randomly adding edges between previously non-adjacent nodes to double the density to 4%, 8%, and 16%. While this approach is based on transformations of existing benchmark instances rather than fully independent random generation, it provides a controlled and systematic framework for analyzing how structural parameters (p_{ij} density and conflict density) affect computational difficulty. Hence, the total number of QKPCG instances is now $45 \times 3 \times 6 = 810$. Information on these 810 instances is provided in Table 2, and all instances are available at [17] to ensure full reproducibility of the experiment.

Table 2. Description of the 18 scenarios with 45 instances per scenario, with all the knapsack capacities set to 1000

Scenarios	n	Conflict graph density	p_{ij} density
Scenario 1 Original 45 instances from [11]	100, 150, 200	2%, 4%, 8%	100%
Scenario 2	100, 150, 200	2%, 4%, 8%	90%
Scenario 3	100, 150, 200	2%, 4%, 8%	75%
Scenario 4	100, 150, 200	2%, 4%, 8%	50%
Scenario 5	100, 150, 200	2%, 4%, 8%	25%
Scenario 6	100, 150, 200	2%, 4%, 8%	0%
Scenario 7	100, 150, 200	1%, 2%, 4%	100%
Scenario 8	100, 150, 200	1%, 2%, 4%	90%
Scenario 9	100, 150, 200	1%, 2%, 4%	75%
Scenario 10	100, 150, 200	1%, 2%, 4%	50%
Scenario 11	100, 150, 200	1%, 2%, 4%	25%
Scenario 12	100, 150, 200	1%, 2%, 4%	0%
Scenario 13	100, 150, 200	4%, 8%, 16%	100%
Scenario 14	100, 150, 200	4%, 8%, 16%	90%
Scenario 15	100, 150, 200	4%, 8%, 16%	75%
Scenario 16	100, 150, 200	4%, 8%, 16%	50%
Scenario 17	100, 150, 200	4%, 8%, 16%	25%
Scenario 18	100, 150, 200	4%, 8%, 16%	0%

Source: created by authors.

In the next section, we will discuss generating solutions for these 810 instances.

4 Gurobi Empirical Results

4.1 Discussion

As mentioned previously, all published solution approaches (through April 2026) for the QKPCG do not guarantee optimal solutions or even provide bounds on how close their solutions are to the optimums [9]. As OR practitioners, using a solution approach that provides no guarantee on solution quality would most likely be difficult to “sell” to management. Alternatively, if faced with an industrial application requiring the solution of a QKPCG, the OR practitioner could consider using a method that guarantees the optimum or at least bounds the solution close to the optimum. However, since no specialized exact solution procedures for the QKPCG exist in the literature and developing such a procedure from “scratch” would be costly, time-consuming, and error-prone, the use of commercial integer-programming software might be preferred. This would be especially true if the OR practitioner’s corporation already was using such software. The downside to using proprietary software such as Gurobi is that the solution process is not transparent to the user. However, the user has the ability to either use default parameter settings and let Gurobi logic determine the best way to adjust the settings based on the given problem or the user can manually adjust the parameters.

The authors of this current paper have been successful at solving more than 60,000 binary integer programming problems using general-purpose integer programming software largely using default parameter settings, but sometimes adjusting the parameters or using the software in an iterative manner. However, using default parameter settings is usually a good starting point. Among these 60,000 instances were a large number of hard 0-1 knapsack problems [16]. Hence, based on these results, it seemed reasonable to try Gurobi with all default parameter values to solve these QKPCG instances (more complicated scenarios could be tried later). Using Gurobi in a default mode to solve an integer programming problem means that either a mipgap value less than 0.0001 is achieved or the maximum time has been reached. For a maximization problem, mipgap is the difference between the best upper bound and the best current solution divided by the best current solution.

The computational strategy adopted in this study is consistent with prior work on large-scale binary integer programming problems, where the use of default solver settings was shown to be an effective and robust baseline. In this paper, this strategy is applied specifically to QKPCG instances

to investigate how problem structure influences solvability. We note that the distinction between “easy” and “hard” instances is not predefined, but emerges empirically based on whether Gurobi is able to obtain proven optimal solutions within the time limit.

In the next subsection, we will report on using Gurobi to solve the 810 QKPCG instances discussed above in two ways. First, we will solve these instances with no initial external solution. This is referred to as a “cold start” for Gurobi. Next, we will use the code for the HSM hybrid search method [9] to generate initial feasible solutions that are used as starting points for Gurobi. This is referred to as a “warm start” for Gurobi.

4.2 Results

All QKPCG instances were solved using Gurobi (11.0.3) and the C++ code provided for HSM by [9] on an AMD Ryzen 5 7600X 6-Core Processor 4.70 GHz with a 64-bit operating system, x64-based processor using Windows 11. All Gurobi default parameter values were used, and the maximum execution time was 1800 seconds.

In Table 3A_o, Table 3A_h, Table 3A_d, Table 3B_o, Table 3B_h, and Table 3B_d, the average objective function values and average execution times are given for the 18 scenarios (results are averaged over 45 QKPCG instances) described in Table 2 using HSM, Gurobi cold start (CS), and Gurobi warm start (WS). The subscripts o, h, and d stand for original conflict graph densities, half conflict graph densities, and double conflict graph densities, respectively. Also, the A tables report on p_{ij} densities of 75%, 90% and 100%. Whereas the B tables report on p_{ij} densities of 0%, 25%, and 50%. Table 3A_o, Table 3A_h, and Table 3A_d also show the average improvements of Gurobi (WS) over Gurobi (CS) objective function values. In Table 4_o, Table 4_h, and Table 4_d, the average mipgap values and number of proven optimums are given for each of the 18 scenarios described in Table 2 using Gurobi CS and Gurobi WS. It can easily be seen from these tables that regardless of the conflict graph densities, the mip gaps are reduced and the number of proven optimums are increased as the p_{ij} densities decrease.

For the original 45 QKPCG instances from [6] that all had p_{ij} densities of 100%, the average HSM objective function value was 1.2% higher (better) than the Gurobi CS objective function value and the average HSM execution time was less than 1% of the Gurobi CS execution time. Although HSM was very fast and gave slightly better solutions than Gurobi CS, there are absolutely no guarantees on

the quality of the HSM solutions. Guarantees on solution quality are very important when solving industrial applications. In contrast, Gurobi CS generated 20 proven optimums for the 45 instances (44%) and bounds (although very loose) for the other 25 instances.

Looking more closely at the Gurobi solutions to these 45 QKPCG instances, we see that optimums were obtained in an average of 136 seconds for the 20 instances in data sets 1qkpcg, 2qkpcg, 3qkpcg, and 4qkpcg. However, for the other 25 instances, no optimums were obtained by Gurobi even after 1800 seconds of execution time. In fact, for these 25 instances, the mipgap values after 1800 seconds of execution time ranged from 10% to 130% with an average of 60%. To show how poor these bounds are, note that to be considered optimum, the mipgap must be less than 0.01%. Remember that all 45 of these instances have p_{ij} densities ($p_{ij} > 0$) of 100%, and that Gurobi was able to find proven optimums within the maximum time of 1800 seconds only for instances with $n = 100$ across all considered conflict graph densities (2%, 4%, 8%) and for instances with $n = 150$ at a conflict graph density of 2%.

However, when we used HSM to warm start Gurobi, the resulting Gurobi solutions were identical to the HSM solutions; however, with Gurobi WS, it is known that 20 of the solutions are optimal (the same as with Gurobi CS). We found it both interesting and disappointing that Gurobi WS was unable to improve upon the HSM solutions. Nevertheless, Gurobi WS was able to reduce the average mipgap value compared to Gurobi CS. When the p_{ij} density is kept at 100%, the results for the half and double conflict graph densities are similar to those for the original 45 QKPCG instances.

As can be seen from Table 3A_o, Table 3A_h, Table 3A_d, Table 4_o, Table 4_h and Table 4_d, when the p_{ij} density is decreased to 90% and 75%, for all three of the conflict graph density scenarios (1/2/4, 2/4/8, 4/8/16), the results are similar to the results for the 100% p_{ij} density cases. Among the 75%, 90%, and 100% p_{ij} density cases, the best results are realized for p_{ij} density at 75% and conflict graph densities of 1/2/4 percent. In this case, the mip gaps are under 10%, and the number of proven optimums is 35 (78%) for both Gurobi CS and Gurobi WS. Overall, the instances summarized in Table 3A_o, Table 3A_h, and Table 3A_d, the Gurobi WS solutions were about 1% better than the Gurobi CS solutions. Although 1% improvement might not seem like a lot, since HSM executes very quickly, it might be advantageous to warm start Gurobi when solving

industrial applications involving QKPCGs that have p_{ij} densities 75% or greater.

Table 3A. Summary average results using HSM, Gurobi cold start (CS), Gurobi warm start (WS) for p_{ij} densities 75%, 90%, 100% with conflict graph density of original 2/4/8

Conflict Graph Density: Original 2/4/8			
p_{ij} densities	Solution method	Obj fn	Time (s)
100%	HSM	16049.2	7
	Gurobi CS	15853.2	1061
	Gurobi WS	16049.2	1070
	Gurobi warm start-improvement	1.2%	
90%	HSM	13334.8	6
	Gurobi CS	13221.9	868
	Gurobi WS	13334.8	870
	Gurobi warm start-improvement	0.9%	
75%	HSM	9760.9	3
	Gurobi CS	9685.2	835
	Gurobi WS	9760.9	834
	Gurobi warm start-improvement	0.8%	

Source: created by the authors.

Table 3A_h. Summary average results using HSM, Gurobi cold start (CS), Gurobi warm start (WS) for p_{ij} densities 75%, 90%, 100% with conflict graph density of half 1/2/4

Conflict Graph Density: Half 1/2/4			
p_{ij} densities	Solution method	Obj fn	Time (s)
100%	HSM	22196.2	11
	Gurobi CS	22006.7	1184
	Gurobi WS	22196.2	1176
	Gurobi warm start-improvement	0.9%	
90%	HSM	18473.2	6
	Gurobi CS	18456.5	860
	Gurobi WS	18473.2	870
	Gurobi warm start-improvement	0.1%	
75%	HSM	13446.2	3
	Gurobi CS	13419.3	497
	Gurobi WS	13446.2	510
	Gurobi warm start-improvement	0.2%	

Source: created by the authors.

However, once the p_{ij} density drops to 50% or lower, Gurobi CS performs essentially as good as Gurobi WS with 402 proven optimums found for these 405 instances (99%). In other words, in this case, there is no benefit to warm-starting Gurobi with the HSM solution. It is interesting to note that

when the p_{ij} density drops to 0%, Gurobi outperforms HSM in terms of both solution quality and execution time. This is not surprising because at 0% p_{ij} density, the problem no longer has a quadratic component. Over the 810 QKPCG instances, Gurobi CS obtained 757 proven optimums (93.5%) and Gurobi WS obtained 758 proven optimums (93.6%).

Table 3A_d. Summary average results using HSM, Gurobi cold start (CS), Gurobi warm start (WS) for p_{ij} densities 75%, 90%, 100% with conflict graph density of double 4/8/16

Conflict Graph Density: Double 4/8/16			
p_{ij} densities	Solution method	Obj fn	Time (s)
100%	HSM	9960.5	5
	Gurobi CS	9766.0	1047
	Gurobi WS	9960.5	1047
	Gurobi warm start-improvement	2.0%	
90%	HSM	8288.8	3
	Gurobi CS	8196.9	974
	Gurobi WS	8288.8	947
	Gurobi warm start-improvement	1.1%	
75%	HSM	6189.2	1
	Gurobi CS	6067.3	1136
	Gurobi WS	6189.2	1123
	Gurobi warm start-improvement	2.0%	

Source: created by the authors.

Table 3B_e. Summary average results using HSM, Gurobi cold start (CS), Gurobi warm start (WS) for p_{ij} densities 0%, 25%, 50% with conflict graph density of original 2/4/8

Conflict Graph Density: Original 2/4/8			
p_{ij} densities	Solution method	Obj fn	Time (s)
50%	HSM	5340.7	2
	Gurobi CS	5340.7	237
	Gurobi WS	5339.5	263
25%	HSM	2556.5	3
	Gurobi CS	2556.5	3
	Gurobi WS	2556.5	3
0%	HSM	1325.9	40
	Gurobi CS	1330.4	5
	Gurobi WS	1330.4	5

Source: created by the authors.

Table 3B_h. Summary average results using HSM, Gurobi cold start (CS), Gurobi warm start (WS) for p_{ij} densities 0%, 25%, 50% with conflict graph density of half 1/2/4

Conflict Graph Density: Half 1/2/4			
p_{ij} densities	Solution method	Obj fn	Time (s)
50%	HSM	7121.8	1
	Gurobi CS	7121.6	87
	Gurobi WS	7121.8	111
25%	HSM	3199.5	2
	Gurobi CS	3199.5	1
	Gurobi WS	3199.5	1
0%	HSM	1392.3	47
	Gurobi CS	1396.3	1
	Gurobi WS	1396.3	1

Source: created by the authors.

Table 3B_d. Summary average results using HSM, Gurobi cold start (CS), Gurobi warm start (WS) for p_{ij} densities 0%, 25%, 50% with conflict graph density of double 4/8/16

Conflict Graph Density: Double 4/8/16			
p_{ij} densities	Solution method	Obj fn	Time (s)
50%	HSM	3507.2	1
	Gurobi CS	3507.2	264
	Gurobi WS	3507.2	269
25%	HSM	1865.1	2
	Gurobi CS	1865.1	4
	Gurobi WS	1865.1	4
0%	HSM	1231.9	27
	Gurobi CS	1236.5	7
	Gurobi WS	1236.5	6

Source: created by the authors.

Table 4_e. Summary average mipgap results and number of proven optimums using Gurobi cold start (CS) and Gurobi warm start (WS) with conflict graph density of original 2/4/8

Conflict Graph Densities: Original 2/4/8			
p_{ij} densities	Solution method	Ave mipgap	Number optimums
100%	Gurobi CS	33.5%	20
	Gurobi WS	31.9%	20
90%	Gurobi CS	28.0%	27
	Gurobi WS	26.3%	27
75%	Gurobi CS	25.8%	25
	Gurobi WS	24.0%	25
50%	Gurobi CS	0.2%	44
	Gurobi WS	0.2%	44
25%	Gurobi CS	0%	45
	Gurobi WS	0%	45
0%	Gurobi CS	0%	45
	Gurobi WS	0%	45

Source: created by the authors.

Table 4_h. Half scenario: summary average mipgap results and number of proven optimums using Gurobi cold start (CS) and Gurobi warm start (WS) with conflict graph density of half 1/2/4

Conflict Graph Density: Half 1/2/4			
p_{ij} densities	Solution method	Ave mipgap	Number optimums
100%	Gurobi CS	19.8%	17
	Gurobi WS	18.4%	19
90%	Gurobi CS	10.8%	27
	Gurobi WS	10.9%	26
75%	Gurobi CS	9.6%	35
	Gurobi WS	9.5%	35
50%	Gurobi CS	0%	45
	Gurobi WS	0%	45
25%	Gurobi CS	0%	45
	Gurobi WS	0%	45
0%	Gurobi CS	0%	45
	Gurobi WS	0%	45

Source: created by the authors.

Table 4_d. Summary average mipgap results and number of proven optimums using Gurobi cold start (CS) and Gurobi warm start (WS) with conflict graph density of double 4/8/16

Conflict Graph Density: Double 4/8/16			
p_{ij} densities	Solution method	Ave mipgap	Number optimums
100%	Gurobi CS	30.8%	22
	Gurobi WS	28.3%	22
90%	Gurobi CS	27.3%	27
	Gurobi WS	25.1%	27
75%	Gurobi CS	42.0%	20
	Gurobi WS	34.8%	20
50%	Gurobi CS	0.3%	43
	Gurobi WS	0.2%	43
25%	Gurobi CS	0%	45
	Gurobi WS	0%	45
0%	Gurobi CS	0%	45
	Gurobi WS	0%	45

Source: created by the authors.

5 Summary and Future work

The quadratic knapsack problem with conflict graphs (QKPCG) was introduced in a 2017 paper [6] as a model for designing and optimizing environmental economic policies in a protective zone. Since the introduction of the QKPCG, all papers in the literature through April 2026 have dealt with solving only the 45 QKPCG test instances that were also introduced in [6]. The solution procedures presented in these papers have focused on methods that do not generate proven optimums. In this paper, the original 45 QKPCG test instances were systematically expanded to a more

comprehensive set of 810 test instances (765 new instances). This benchmark set is publicly available at [17].

and will allow researchers to test how sensitive the performance of QKPCG solution methods is to changes in conflict graph density and p_{ij} density. Next, the benefit of using solutions generated by a hybrid search method (HSM) from the literature as input to Gurobi when the p_{ij} densities are 75% or greater was demonstrated. However, when the p_{ij} densities are less than 75%, the use of HSM solutions as input to Gurobi was shown to be of no benefit. Over the 810 test instances, Gurobi determined proven optimums for over 93% of them (99% when the density of the quadratic profit terms is less than 75%). The primary contribution of this paper is not the development of a new algorithm, but rather the first systematic investigation of when QKPCG instances can be solved to proven optimality using a general-purpose solver. These results can be particularly helpful to OR practitioners using Gurobi to solve such problems. It is important to note that these results were achieved using all Gurobi default parameter settings. In future work, we plan to use Gurobi's tuning tool to try to improve Gurobi's performance on the QKPCG. Also, the use of Gurobi in an iterative manner to solve QKPCG instances will be explored.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, the author(s) utilized artificial intelligence (AI) technologies solely for the purpose of linguistic refinement and grammatical correction. Following the application of these tools, the author(s) comprehensively reviewed, verified, and edited the resulting text. The author(s) maintain full accountability and responsibility for the accuracy, integrity, and final content of the publication.

References:

- [1] Gallo, G., Hammer, P.L., Simeone, B. Quadratic knapsack problems, Math. Program. Stud. 1980, 12: 132–149. <https://doi.org/10.1007/BFb0120892>.
- [2] Pisinger, D. The quadratic knapsack problem — a survey. Discrete Appl. Math. 2007, 155: 623–648. <https://doi.org/10.1016/j.dam.2006.08.007>.
- [3] Cacchiani, V., Lori, M., Locvatelli, A., Martello, S. Knapsack problems-an overview of recent advances. Part II: multiple, multidimensional, and quadratic knapsack

- problems. *Computers & Operations Research*. 2022, 143: 105693. <https://doi.org/10.1016/j.cor.2021.105693>.
- [4] Fomeni, F.D., Kaparis, K., Letchford, A.N. A cut-and-branch algorithm for the Quadratic Knapsack Problem. *Discrete Optimization*. 2022, 44: 1-18. <https://doi.org/10.1016/j.dispot.2020.100579>.
- [5] Fomeni, F.D. A lifted-space dynamic programming algorithm for the Quadratic Knapsack Problem. *Discrete Applied Mathematics*. 2023, 335: 52-68. <https://doi.org/10.1016/j.dam.2023.02.003>.
- [6] Shi, X., Wu, L., & Meng, X. (2017). A new optimization model for the sustainable development: Quadratic knapsack problem with conflict graphs. *Sustainability*, 9(2), 236. <https://doi.org/10.3390/su9020236>.
- [7] Dahmani, I., Hifi, M., Saadi, T., & Yousef, L. (2020). A swarm optimization-based search algorithm for the quadratic knapsack problem with conflict graphs, *Expert Systems with Applications*, 148, 113224. <https://doi.org/10.1016/j.eswa.2020.113224>.
- [8] Dahmani, I., & Hifi, M. (2021). A modified descent method-based heuristic for binary quadratic knapsack problems with conflict graphs, *Annals of Operations Research*, 298(1-2), 125-147. <https://doi.org/10.1007/s10479-019-03290-3>.
- [9] Zhou, Q., Hao, J-K., Jiang, Z-Z., Wu, Q. (2024). An effective hybrid search method for the quadratic knapsack problem with conflict graphs, *Journal of the Operational Research Society*, 75(3), 1000-1010. <https://doi.org/10.1080/01605682.2023.2228339>.
- [10] Cacchiani, V., Lori, M., Locvatelli, A., Martello, S. (2022a). Knapsack problems-an overview of recent advances, Part I: single knapsack problems. *Computers & Operations Research*, 143: 105692. <https://doi.org/10.1016/j.cor.2021.105692>.
- [11] Billionnet, A., & Soutif, E. (2004). An exact method based on Lagrangian decomposition for the quadratic knapsack problem. *European Journal of Operational Research*, 157(3), 565-575. [https://doi.org/10.1016/S0377-2217\(03\)00244-3](https://doi.org/10.1016/S0377-2217(03)00244-3).
- [12] Cunha, J.O., Simonetti, L., Lucena, A. (2016) Lagrangian heuristics for the Quadratic Knapsack Problem, *Comput Optim Appl*. 63: 97-120. <https://doi.org/10.1007/s10589-015-9763-3>.
- [13] Galli, L., Martello, S., Rey, C., Toth, P. (2025). The Quadratic Knapsack Problem with Setup, *Computers & Operations Research*, 173: 106873. <https://doi.org/10.1016/j.cor.2024.106873>.
- [14] Smith-Miles, K., & Bowly, S. (2015). Generating new test instances by evolving in instance space, *Computers & Operations Research*, 63, 102-113. <http://dx.doi.org/10.1016/j.cor.2015.04.022>.
- [15] Smith-Miles, K., & Muñoz, M. A. (2023). Instance space analysis for algorithm testing: Methodology and software tools, *ACM Computing Surveys*, 55(12), 1-31. <http://dx.doi.org/10.1145/3572895>.
- [16] Song, M.S., Lin, P.H., Lu, Y., Shively-Ertas, E., Vasko, F. J. (2025a) A practical approach for dealing with hard knapsack problems using general-purpose integer programming software, *International Transactions in Operational Research*, 32, 1472-1493. <https://doi.org/10.1111/itor.13449>
- [17] GitHub Platform, [Online]. <https://github.com/AmberRK/QKPCG/tree/main> (Accessed Date: March 19, 2026).

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

Amber Kulp, Myung Soon Song, Francis J. Vasko carried out the conceptualization, methodology validation, formal analysis, investigation, writing—review and editing, and visualization.

Amber Kulp controlled and executed the software, data curation, and resources.

Francis J. Vasko, Yun Lu writing—original draft preparation.

Yun Lu, Francis J. Vasko supervision.

Yun Lu project administration.

All authors have read and agreed to the published version of this paper.

Sources of Funding for Research Presented in Determining when Gurobi can Optimally Solve the Quadratic Knapsack Problem with Conflict Graphs

No funding was received for conducting this study.

Conflict of interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0

https://creativecommons.org/licenses/by/4.0/deed.en_US