sustainability

MDPI

Article

A New Optimization Model for the Sustainable Development: Quadratic Knapsack Problem with Conflict Graphs

Xiaochuan Shi [1,†], Lei Wu [2,†] and Xiaoliang Meng [1,†,*]

1 International School of Software, Wuhan University, 37 Luoyu Road, Wuhan 430079, China; shixiaochuan@whu.edu.cn
2 Wenlan School of Business, Zhongnan University of Economics and Law, 182 Nanhu Avenue, Wuhan 430073, China; wu.lei@zuel.edu.cn
* Correspondence: xmeng@whu.edu.cn; Tel.: +86-27-6877-1236
† These authors contributed equally to this work.

Academic Editors: Yichun Xie, Xinyue Ye and Clio Andris
Received: 12 December 2016; Accepted: 4 February 2017; Published: 9 February 2017

Abstract: New information technology constantly improves the efficiency of social networks. Using optimization and decision models in the context of large data sets attracts extensive attention. This paper investigates a novel mathematical model for designing and optimizing environmental economic policies in a protection zone. The proposed model is referred to as the quadratic knapsack problem with conflict graphs, which is a new variant of the knapsack problem family. Due to the investigated problem processing a high complex structure, in order to solve efficiently the problem, we develop a metaheuristic which is based on the large neighborhood search. The proposed method embeds a construction procedure into a sophistical neighborhood search. For more details, the construction procedure takes charge of finding a starting solution while the investigated neighborhood search is used to generate and explore the solution space issuing from the provided starting solution. In order to highlight our theoretical model, we evaluate the model on a set of complex benchmark data sets. The obtained results demonstrate that the investigated algorithm is competitive and efficient compared to legacy algorithms.

Keywords: Modelization; quadratic; conflict; sustainable development

1. Introduction

Nowadays, the evolution of information provides us the opportunity to design reasonable policies through the use of large data sets (see, e.g., Lee et al. [1]). Such technologies constantly improve the efficiency of the social network. Using optimization and decision models in social planning becomes a big challenge (see, e.g., Lee, Kim and Kim [2], Li and Lin [3] , Shim and Park [4]). A computational model is presented in this paper to design environmental economic policies for sustainable development. As the global ecological and environmental problems become increasingly severe, people's lives and social development face serious constraints and impact. Governments have proposed the need of the policies for protection zones, such as China's Ecological Red-line policies, to protect the regional economic and social development, and regional ecological safety. We propose the hypothesis of using a new model to optimize the policies through the use of large data sets from stakeholders' services and public's crowdsourcing. Such a model can be considered as a new generation from the knapsack problem family: the *Quadratic Knapsack Problem with Conflict Graphs* (QKPCG). An instance of QKPCG is composed of a knapsack of a capacity c, a set I of n items, and a set E of incompatible couples of items (i.e., $E \subseteq \left\{(i,j) \in I \times I,\ \ i < j\right\}$). Each item $i \in I$, it is associated

with a positive weight w_i and a positive profit p_i. For each compatible couple of items (i, j), where $(i, j) \notin E$ and $i < j$, it is associated with a positive profit p_{ij}, which implies that an additional profit occurs when both of items i and j are selected to be placed into the knapsack. The objective of QKPCG consists of maximizing the total profit of items selected to be placed into the knapsack under the capacity constraint, where all selected items must be compatible. The quadratic program related to QKPCG can be defined as Equations (1)–(3):

$$
\begin{aligned}
(\mathrm{QP}_{QKPCG}) \quad \max \quad & \textstyle\sum_{i\in I} p_i x_i + \sum_{i\in I} \sum_{j\in I, j>i} p_{ij} x_i x_j && && (1)\\
\text{s.t.} \quad & \textstyle\sum_{i\in I} w_i x_i \leq c && && (2)\\
& x_i + x_j \leq 1 && \forall\, (i,j) \in E && (3)\\
& x_i \in \{0,1\} && \forall\, i \in I. &&
\end{aligned}
$$

The decision variable $x_i, \forall\, i \in I$, equals to 1 if the i-th item is selected; 0 otherwise. Inequality (2) represents the capacity constraint and Inequalities (3) denote the disjunctive constraints, which ensure that compatibility of all items belonging to the knapsack.

On the background of sustainable development, the program of QKPCG (QP_{QKPCG}) can be used to establish a reasonable economic development strategy in the environmental protection zone. In QP_{QKPCG}, the objective is to optimize the gain related to the development planning with limited natural resources without undermining the stability of natural biotic systems. More preciously, assume that a development planning is composed of n different components, where each item of QKPCG can be considered as a component. The related decision variable $x_i = 1$ if the ith component is applied; otherwise $x_i = 0$. A component of the planning can be defined as follows: a natural resource is used in an investment. Therefore, the objective function (cf., Formula (1)) measures the total reward related to all performed investments while the capacity constraint (cf., Inequality (2)) ensures that the effect by the exploitation of natural resources cannot exceed the fixed ecological boundary. Furthermore, the quadratic term in the objective function (i.e., $(p_{ij}x_i x_j)$) represents the additional profit by combining the investment i and j. In order to make our model more realistic, we introduce the disjunctive constraints (cf., Inequalities (3)). A disjunctive constraint simulates the fact that two different investments cannot share the same natural resource.

QKPCG is an NP-hard problem (see, e.g., Garey and Johnson [5]). It reduces to the Quadratic Knapsack Problem (QKP) (see, e.g., [6,7]) when the constraints (3) are dropped and to the Knapsack Problem with Conflict Graphs (KPCG) (see, e.g., Pferschy and Schauer [8]) when $p_{ij} = 0$, for all $i \neq j = 1, \ldots, n$. Note that, in literature, the KPCG is also named by Yamada, Kataoka and Watanabe [9] as the Knapsack Problem with Conflict Graphs (KPCG). Due to the complexity of the QKP and the KPCG, few studies on exact methods for these two problems have been realized in the literature. Recently, most results on these topics are based on meteheuristics, which focus on providing high quality solution for large complex cases (see, e.g., [10–12]).

For the rest of the paper, Section 2 discusses the starting solution procedure, which provides an initial solution for QP_{QKPCG}. Section 3 introduces an efficient metaheuristic for improving the solution at hand. Section 4 evaluates the performance of the proposed approach on a group of benchmark data sets. Finally, the contents of the paper are summarized in Section 5.

2. A Starting Solution Procedure for QKPCG

This section describes a starting solution procedure, noted by 2PH, which is based on solving successively QP_{QKPCG}. The used procedure is composed of two phases, where the first phase aims at determining a feasible solution and the second phase tries to improve the provided solution by exploring its neighborhood. Such a procedure has also been used in [11,12] for approximately solving the KPCG. Unless noted otherwise, we assume that all items are sorted descending by their ratio of profit per weight (i.e., $p_i/w_i, \forall\, i = 1, \ldots, n$).

2.1. An Integer Linear Programming of the QKPCG

To reduce the computational effort caused by the quadratic term, we apply a classic linearization of the QKPCG proposed in Glover and Woolsey [13]. The considered integer linear program associated with the QKPCG, notated by ILP_{QKPCG} can be defined as follows.

$$
\begin{array}{lll}
(\text{ILP}_{QKPCG}) \quad \max & \sum_{i\in I} p_i x_i + \sum_{i\in I, j\in I, j>i} p_{ij} y_{ij} & \\
\text{s.t.} & \sum_{i\in I} w_i x_i \leq c & (4)\\
& x_i + x_j \leq 1 \qquad \forall\, (i,j) \in E & (5)\\
& x_i + x_j \leq 1 + y_{ij} \qquad \forall\, i \in I, j \in I, j > i & (6)\\
& y_{ij} \leq x_i \; and \; y_{ij} \leq x_j \qquad \forall\, i \in I, j \in I, j > i & (7)\\
& x_i \in \{0,1\}, y_{ij} \in \{0,1\} \qquad \forall\, i \in I, j \in I, j > i. &
\end{array}
$$

In view of complexity of ILP_{QKPCG}, in order to produce efficient solutions, 2PH consists of determining a feasible solution of the QKPCG by considering successively the quadratic term and the constraints (4)–(7).

2.2. The First Phase

At the first phase, an instance of QKPCG is reduced to an instance of KPCG, where the quadratic term is not taken account at the current step. The corresponding integer linear program of KPCG can be formally written as follows:

$$
\begin{array}{lll}
(\text{ILP}_{KPCG}) \quad \max & \sum_{i\in I} p_i x_i & \\
\text{s.t.} & \sum_{i\in I} w_i x_i \leq c & (8)\\
& x_i + x_j \leq 1 \qquad \forall\, (i,j) \in E & (9)\\
& x_i \in \{0,1\} \qquad \forall\, i \in I. &
\end{array}
$$

From QP_{QKPCG}, we can observe that a feasible solution of KPCG is also feasible for QKPCG. This is due to the fact that, in ILP_{KPCG}, we ignore only the additional profit when choosing two items but respect the capacity constraint (8) and the disjunctive constraints (9). Therefore, a feasible solution of QKPCG can be computed by solving successively two optimization problems: a *weighted independent set problem* (see ILP_{WIS}), extracted from ILP_{KPCG} with the elimination of the capacity constraint (8); a *binary knapsack problem* (see ILP_K) related to the independent set by solving ILP_{WIS}. ILP_{WIS} is first solved by providing an independent set. Secondly, ILP_K is solved by computing a feasible solution of ILP_{KPCG}, which is also feasible for QP_{QKPCG}, with the consideration of the independent set yielded at the first phase. Let IS ($IS \subseteq I$) be a feasible solution of ILP_{WIS}, then the linear programs referring to ILP_{WIS} and ILP_K can be defined by following expressions:

$$
(\text{ILP}_{WIS}) \begin{cases} \max & \sum_{i\in I} p_i x_i \\ \text{s.t.} & x_i + x_j \leq 1,\ \forall (i,j) \in E \\ & x_i \in \{0,1\},\ \ \forall\, i \in I, \end{cases} \qquad (\text{ILP}_K) \begin{cases} \max & \sum_{i\in IS} p_i x_i \\ \text{s.t.} & \sum_{i\in IS} w_i x_i \leq c, \\ & x_i \in \{0,1\},\ \forall\, i \in IS. \end{cases}
$$

Algorithm 1 displays the general idea to compute an available solution of ILP_{KPCG}. At Step 1, IS is initialized as a null set, which is an evident solution of ILP_{WIS}. The loop from Step 2 to Step 6 serves to produce IS iteratively with the un-selected items included in I. For each iteration, the item with the highest ratio of profit per weight is selected to be added into IS, while the selected item and its incompatible partners are immediately dropped from I. The loop stops if there are no more available items that can be included into IS. Steps 7–12 are used to compute a feasible solution for ILP_{KPCG}. One first checks whether the current solution IS respects all constraints of ILP_{KPCG}. If IS respects the

capacity constraint (8) of ILP_{KPCG}, IS is also feasible for KPCG and further for QKPCG. Otherwise, ILP_K related to IS is solved to produce a solution ensuring the capacity constraint (8). To achieve the best performance of the investigated approach, ILP_K is solved by applying the exact algorithm elaborated by Martello, Pisinger and Toth [14].

Algorithm 1 Determine a feasible solution for ILP_{KPCG}

Input: An instance I of ILP_{KPCG}.
Output: A feasible solution S_{KPCG} for ILP_{KPCG}.
1: **Initialization:** Set IS as empty set and $I = \{1, \ldots, n\}$;
2: **while** I is not empty **do**
3: Set $i = \text{argmax}\left\{ \frac{p_i}{w_i} \mid \frac{p_i}{w_i} \geq \frac{p_k}{w_k},\ k \in I \right\}$;
4: Add i into IS: $IS = IS \cup \{i\}$;
5: Eliminate i with all its incompatible items j such that $(i, j) \in E$ from I;
6: **end while**
7: Generate ILP_K defined on the items of IS;
8: **if** IS respect the capacity constraint (8) of ILP_{KPCG} **then**
9: Set S_{KPCG} as IS;
10: **else**
11: Set S_{KPCG} as the optimal solution of ILP_K;
12: **end if**
13: **return** S_{KPCG}.

2.3. The Second Phase

Note that, at the first phase, an instance of QKPCG is reduced to an instance of KPCG, where the quadratic term is instantly ignored. Therefore, the provided solution might be poor for QKPCG. This is due to the fact that one does not take into consideration optimization of the objective function of ILP_{QKPCG}. Therefore, the aim of the second phase is to improve the quality of the solution obtained at the first phase (i.e., the solution S_{KPCG} yielded by Algorithm 1) with help from an iterative local search. The applied local search serves to ameliorate a given solution by alternatively performing a *building* procedure and an *exploring* procedure. The *building procedure* generates a k-neighborhood of S_{KPCG} by cleaning the values of k fixed variables of the solution vector related to S_{KPCG}. The *exploring procedure* explores the generated neighborhood, which can be viewed as reduction of the original solution space, for determining a local optimum solution. The second phase stops when no valid improvement occurs.

Algorithm 2 shows how the proposed local search operates. Let S_{QKPCG} be the solution by applying Algorithm 1 and set α as a constant number. The loop from Step 2 to Step 6 of the local search alternatively generates and explores a series of neighborhoods in order to ameliorate the starting solution. At Step 3, the current best solution, namely $S^{\star}_{QKPCG}$, is replaced by S_{QKPCG} if it has a better value. Step 4 cleans the values of α variables in S_{QKPCG}, where the items with the highest degree are favored. For an item, the value of its degree is the number of incompatible items related to this item. Let i be the item of the highest degree, where $x_i = 1$ in S_{QKPCG}. For each iteration, set x_i and its incompatible partners as free. The incompatible items related to x_i can be noted by x_j, such that $(i, j) \in E$ and $(j, k) \notin E$, where k ($k \neq i$) denotes the index of variables fixed to 1 in S_{QKPCG}. At Step 5, S_{QKPCG} is updated with the local optimum solution by exploring the current neighborhood, where the neighborhood corresponds to a subproblem of the S_{QKPCG}. In other words, ILP_{QKPCG} is solved on a reduced instance of QKPCG. Finally, Algorithm 2 stops when no better solution can be achieved and returns the best solution $S^{\star}_{QKPCG}$ deduced from the starting solution.

Algorithm 2 A local search for improving a given solution

Input: S_{QKPCG}, a feasible solution of the QKPCG.
Output: $S^{\star}_{QKPCG}$, a local optimum solution of the QKPCG.
1: Initialize $S^{\star}_{QKPCG}$ as an evident solution, where all variables equal to 0;
2: **while** S_{QKPCG} is better than $S^{\star}_{QKPCG}$ **do**
3: Set $S^{\star}_{QKPCG}$ as S_{QKPCG};
4: Generate a neighborhood of S_{QKPCG} by cleaning α fixed variables;
5: Explore the current neighborhood to compute a local optimum solution: S_{QKPCG};
6: **end while**
7: **return** $S^{\star}_{QKPCG}$.

3. A Neighborhood Search-Based Metaheuristic

In the domain of operations research, the neighborhood search is a common technique used to improve a given feasible solution or to correct a given infeasible solution. A special case of the neighborhood search, named the Large Neighborhood Search (LNS), was first introduced by Shaw [15] for solving large-scale instances of the vehicle routing problem. Similar to the local search, LNS is based on the concept of *building* and *exploring* neighborhoods. The difference between the two approaches is that, using the descent method may lead the solution procedure to stagnate in certain local optima; however, using LNS increase the probability for providing better solutions with exploration of a series of more promising solution spaces. This is because, for the local search stated in Section 2.3 (cf., Algorithm 2), either *building* or *exploring* adopts a mono-criterion for removing or inserting items. In order to enlarge the opportunity of escaping from the current local optimum, a random building strategy, which is based on removing items by considering their values of profit per weight, is applied. Algorithm 3 for determining a random neighborhood from a given solution, and Algorithm 4 summarizes the global solution procedure for computing a high quality solution of ILP_{QKPCG}.

Algorithm 3 Remove $\beta|I|$ variables of S_{QKPCG}

Input: S_{QKPCG}, a starting solution of ILP_{QKPCG}.
Output: An independent set IS and a reduced instance I^r of ILP_{QKPCG}.
1: Set $cnt = 0$, $I^r = \emptyset$ and IS to the independent set related to S_{QKPCG};
2: Sort the items of IS in increasing order of their values of profit per weight;
3: **while** $cnt < \beta|I|$ **do**
4: Let r be a random real number varied in $[0, 1]$ and $i = |IS| \times r^{\gamma}$;
5: Remove i^{th} item from IS, i in I^r and set $cnt = cnt + 1$;
6: **for** all j such that $(i, j) \in E$ **do**
7: **if** item j is compatible with all items of IS **then**
8: Add j in I^r and set $cnt = cnt + 1$;
9: **end if**
10: **end for**
11: **end while**
12: **return** IS and I^r.

The *building* procedure described in Algorithm 3 consists of randomly removing at least $\beta|I|$ fixed variables from the current solution set. Consequently, Algorithm 3 aims at building a reduced problem of an instance of ILP_{QKPCG}, which is defined on the set of removed variables. In contrast to the deterministic building procedure applied in the local search (see Algorithm 2), the random building procedure (cf. Algorithm 3) works by randomly exploring the solution space of ILP_{QKPCG}. The triple parameters (β, r, γ) control the variety of sub solution spaces generated by Algorithm 3.

For example, if the values of β and r^{γ} are both small (resp. large), one has little (resp. great) chance of visiting a diversified solution space.

Algorithm 4 A neighborhood search-based metaheuristic

Input: An instance I of ILP$_{QKPCG}$.
Output: $S^{\star}_{QKPCG}$, a local optimum of ILP$_{QKPCG}$.
1: Apply successively Algorithms 1 and 2 to compute a feasible solution of QKPCG, noted by S_{QKPCG};
2: Set $S^{\star}_{QKPCG}$ as an evident solution, where all variables equal to 0;
3: **while** the limit of runtime NSBM$_{time}$ is not attained **do**
4: Apply Algorithm 3 to determine IS and I^{r} associated to S_{QKPCG};
5: Apply Algorithm 1 with I^{r} to compute a new solution S_{QKPCG};
6: Apply Algorithm 2 to determine S_{QKPCG};
7: Update $S^{\star}_{QKPCG}$ if S_{QKPCG} is better;
8: **end while**
9: **return** $S^{\star}_{QKPCG}$.

The complete version of the investigated neighborhood search-based metaheuristic is described in Algorithm 4. It first applies successively Algorithms 1 and 2 to determine an initial solution for ILP$_{QKPCG}$. Then it applies iteratively Algorithms 3, 1 and 2 to explore randomly the solution space of ILP$_{QKPCG}$. Finally, when the runtime limit is met, Algorithm 4 returns the best solution at hand.

4. Simulation Results

For the sustainable development in an environmental protection zone, the designing and optimization of economic policies and key restraining factors were always analyzed with the method of multi-index comprehensive evaluation. The index system was composed of several aspects, i.e., ecological economy, eco-environment, eco-living, ecological culture, ecological institution and so on. From index systems built by different researchers, government planning and reports, and crowdsourcing focuses, more factors would be listed to indicate the level of the aspects. The problem of computing the index systems can be modeled as the investigated Neighborhood Search-Based Metaheuristic (NSBM). In order to conduct the simulation of computing the large data of index systems, this section examines the performance of the investigated NSBM on a group of benchmark data sets, notated as $1qkpcg$–$9qkpcg$. These instances are generated by taking account of the structure used by Billionnet and Soutif [16] for the quadratic terms and the structure used by Yamada, Kataoka and Watanabe [9] for the disjunctive constraints. The program of NSBM is coded in C++ and all tests are performed on an Intel Core i5 with 3.1 Ghz.

The proposed dataset contains 45 instances with different numbers of items and densities. The number of items varies in $\{100, 150, 200\}$ while the density of the graph varies in $\{2\%, 4\%, 8\%\}$. The structure of the new generated benchmark problems is described in Table 1.

Table 2 displays the set of values chosen for parameters in our experiment, which guarantees the best performance for NSBM on the adopted benchmark problems.

Table 1. Description of the benchmark instances.

Class	n	Density	Class	n	Density	Class	n	Density
1qkpcg	100	2%	4qkpcg	150	2%	7qkpcg	200	2%
2qkpcg	100	4%	5qkpcg	150	4%	8qkpcg	200	4%
3qkpcg	100	8%	6qkpcg	150	8%	9qkpcg	200	8%

Table 3 shows the objective values achieved by NSBM and GLPK (version 4.60). Column 1 of Table 3 displays the names of the data sets. In Column 2, we report the objective values of the best

solutions provided by GLPK in 3600 s, noted by V_{GLPK}. Column 3–11 reports respectively the solution information when using different runtime limits (i.e., 50, 100 and 200 s), where Column V_{Mean} (resp. V_{Best}) denotes the mean (resp. maximum) objective value achieved by NSBM over five trials within the corresponding time limit and Column T_{Best} displays the runtime used to reach the best objective value. Table 4 displays percentage improvement in quality of the objective value.

Table 2. Parameters setting used to perform NSBM.

Parameters	α	β	γ	**NSBM$_{time}$ in Seconds**
values	20	15	20	$\{50, 100, 200\}$

Table 3. Performance of NSBM vs GLPK on the benchmark instances.

Instance	V_{GLPK}	**NSBM**								
		$t = 50$			$t = 100$			$t = 200$		
		V_{Mean}	V_{Best}	T_{Best}	V_{Mean}	V_{Best}	T_{Best}	V_{Mean}	V_{Best}	T_{Best}
1qkpcg1	15,482	15,942.8	16,274	0.82	16,091.2	16,338	82.71	16,280.2	16,642	117.78
1qkpcg2	11,774	11,631.4	11,925	42.14	11,862.6	12,218	86.28	12,006.2	12,272	125.34
1qkpcg3	15,129	15,299.2	15,678	9.64	15,312	15,678	9.58	15,612.6	15,843	148.14
1qkpcg4	17,151	19,558.2	19,622	13.30	19,578.4	19,622	47.16	19,594.2	19,659	102.30
1qkpcg5	15,147	15,217.4	15,499	37.25	15,326.8	15,499	37.44	15,378.8	15,499	37.18
2qkpcg1	9018	10,601	10,881	28.13	10,801.6	11,205	78.50	10,895	11,205	82.78
2qkpcg2	9150	10,227.8	10,437	1.89	10,313	10,565	88.88	10,438.2	10,565	88.83
2qkpcg3	10,002	10,941.8	11,163	47.09	11,046.6	11,163	45.12	11,084.8	11,163	47.02
2qkpcg4	12,589	15,427.8	15,599	11.53	15,427.8	15,599	11.54	15,599	15,599	156.07
2qkpcg5	10,810	12,528.4	12,580	20.60	12,541.6	12,580	19.17	12,541.6	12,580	19.28
3qkpcg1	5843	7173.4	7455	17.75	7173.4	7455	18.19	7337	7455	17.00
3qkpcg2	6068	7150	7343	44.46	7248	7343	62.35	7343	7343	57.44
3qkpcg3	6091	7184	7258	7.69	7200.6	7258	82.97	7200.6	7258	88.58
3qkpcg4	7978	7987	7987	18.98	7988.6	7991	64.64	7988.6	7991	59.85
3qkpcg5	6290	7156.4	7350	4.37	7156.4	7350	4.30	7156.4	7350	4.38
4qkpcg1	18,025	18,625.2	19,457	31.32	18,661	19,457	31.32	18,931.4	19,457	29.84
4qkpcg2	17,049	18,075.2	18,702	45.37	18,480	19,240	98.30	18,749	19,240	97.11
4qkpcg3	15,835	20,017.2	20,555	5.91	20,426	20,864	73.57	20,538.2	20,864	73.48
4qkpcg4	26,428	30,259.8	31,082	3.19	30,259.8	31,082	3.18	30,737.8	31,082	3.01
4qkpcg5	15,768	17,455.2	18,435	22.74	17,614.8	18,435	21.57	18,083.4	18,727	124.47
5qkpcg1	10,663	14,618	14,850	0.83	14,625.8	14,850	0.90	14,745	14,871	110.44
5qkpcg2	10,720	14,739.8	15,068	1.80	14,887.2	15,068	1.77	14,956.2	15,068	1.62
5qkpcg3	10,846	14,706.6	14,825	37.73	14,892.4	14,939	65.50	14,946.2	15,132	161.43
5qkpcg4	13,639	18,028.4	18,380	38.23	18,143.6	18,380	37.98	18,272.2	18,380	161.64
5qkpcg5	10,883	15,265.2	15,607	44.82	15,292.4	15,607	42.02	15,313.2	15,607	44.84
6qkpcg1	6032	8699	8700	1.46	8769	8969	71.01	8840	8969	136.26
6qkpcg2	6041	9053.6	9193	18.41	9053.6	9193	18.46	9098.8	9193	18.43
6qkpcg3	6429	8390.8	8499	49.92	8448.8	8508	82.44	8504.4	8508	79.86
6qkpcg4	7366	8540.6	8839	43.04	8540.6	8839	47.29	9156.4	9630	147.43
6qkpcg5	5866	9226.6	9335	2.48	9260.2	9335	60.41	9335	9335	141.68
7qkpcg1	18,900	22,867	24,138	21.45	23,816.2	24,196	79.60	24,187.8	24,602	127.89
7qkpcg2	22,545	27,484.8	28,236	24.87	27,968	28,521	78.41	28,354.6	28,686	152.25
7qkpcg3	15,637	21,855	23,420	43.47	22,252.2	23,420	43.05	22,444.6	23,420	43.48
7qkpcg4	27,146	31,940.2	32,641	23.85	32,291.6	32,641	22.86	32,350.2	32,641	22.36
7qkpcg5	18,714	21,327.2	22,132	44.82	21,605.2	22,735	91.67	21,762.4	22,735	90.17
8qkpcg1	12,388	18,583	18,583	10.49	18,583	18,583	10.12	18,583	18,583	10.45
8qkpcg2	13,221	19,336.6	19,553	30.06	19,336.6	19,553	33.08	19,406.2	19,620	126.94
8qkpcg3	11,054	16,704.8	16,974	13.59	16,879	17,469	62.74	16,998.6	17,469	59.92
8qkpcg4	13,060	20,235	20,513	7.33	20,291.4	20,513	7.31	20,524	20,715	100.65
8qkpcg5	12,960	17,829.2	18,480	40.61	17,829.2	18,480	39.29	18,000.4	18,480	38.88
9qkpcg1	6823	11,157.6	11,180	30.00	11,157.6	11,180	32.71	11,180	11,180	110.27
9qkpcg2	7673	10,386.2	10,470	21.75	10,519	10,577	65.74	10,534.2	10,577	63.77
9qkpcg3	5291	10,609.2	10,869	12.30	10,728.8	10,869	12.30	10,852.6	10,869	12.30
9qkpcg4	8241	10,967	11,238	50.00	10,992.2	11,238	47.31	11,308	11,751	150.00
9qkpcg5	6792	10,597.8	10,713	48.24	10,602.4	10,713	45.96	10,602.4	10,713	47.27
Av.	12,012.38	14,924.63	15,282.62	23.92	15,050.58	15,362.62	45.93	15,194.50	15,433.96	80.89

Table 4. Percentage improvement of NSBM vs GLPK on the benchmark instances.

Instance	V_{GLPK}	NSBM					
		$t = 50$		$t = 100$		$t = 200$	
		IMP_{Mean}	IMP_{Best}	IMP_{Mean}	IMP_{Best}	IMP_{Mean}	IMP_{Best}
1qkpcg1	15,482	2.98%	5.12%	3.93%	5.53%	5.16%	7.49%
1qkpcg2	11,774	−1.21%	1.28%	0.75%	3.77%	1.97%	4.23%
1qkpcg3	15,129	1.12%	3.63%	1.21%	3.63%	3.20%	4.72%
1qkpcg4	17,151	14.04%	14.41%	14.15%	14.41%	14.25%	14.62%
1qkpcg5	15,147	0.46%	2.32%	1.19%	2.32%	1.53%	2.32%
2qkpcg1	9018	17.55%	20.66%	19.78%	24.25%	20.81%	24.25%
2qkpcg2	9150	11.78%	14.07%	12.71%	15.46%	14.08%	15.46%
2qkpcg3	10,002	9.40%	11.61%	10.44%	11.61%	10.83%	11.61%
2qkpcg4	12,589	22.55%	23.91%	22.55%	23.91%	23.91%	23.91%
2qkpcg5	10,810	15.90%	16.37%	16.02%	16.37%	16.02%	16.37%
3qkpcg1	5843	22.77%	27.59%	22.77%	27.59%	25.57%	27.59%
3qkpcg2	6068	17.83%	21.01%	19.45%	21.01%	21.01%	21.01%
3qkpcg3	6091	17.94%	19.16%	18.22%	19.16%	18.22%	19.16%
3qkpcg4	7978	0.11%	0.11%	0.13%	0.16%	0.13%	0.16%
3qkpcg5	6290	13.77%	16.85%	13.77%	16.85%	13.77%	16.85%
4qkpcg1	18,025	3.33%	7.94%	3.53%	7.94%	5.03%	7.94%
4qkpcg2	17,049	6.02%	9.70%	8.39%	12.85%	9.97%	12.85%
4qkpcg3	15,835	26.41%	29.81%	28.99%	31.76%	29.70%	31.76%
4qkpcg4	26,428	14.50%	17.61%	14.50%	17.61%	16.31%	17.61%
4qkpcg5	15,768	10.70%	16.91%	11.71%	16.91%	14.68%	18.77%
5qkpcg1	10,663	37.09%	39.27%	37.16%	39.27%	38.28%	39.46%
5qkpcg2	10,720	37.50%	40.56%	38.87%	40.56%	39.52%	40.56%
5qkpcg3	10,846	35.59%	36.69%	37.31%	37.74%	37.80%	39.52%
5qkpcg4	13,639	32.18%	34.76%	33.03%	34.76%	33.97%	34.76%
5qkpcg5	10,883	40.27%	43.41%	40.52%	43.41%	40.71%	43.41%
6qkpcg1	6032	44.21%	44.23%	45.37%	48.69%	46.55%	48.69%
6qkpcg2	6041	49.87%	52.18%	49.87%	52.18%	50.62%	52.18%
6qkpcg3	6429	30.51%	32.20%	31.42%	32.34%	32.28%	32.34%
6qkpcg4	7366	15.95%	20.00%	15.95%	20.00%	24.31%	30.74%
6qkpcg5	5866	57.29%	59.14%	57.86%	59.14%	59.14%	59.14%
7qkpcg1	18,900	20.99%	27.71%	26.01%	28.02%	27.98%	30.17%
7qkpcg2	22,545	21.91%	25.24%	24.05%	26.51%	25.77%	27.24%
7qkpcg3	15,637	39.76%	49.77%	42.30%	49.77%	43.54%	49.77%
7qkpcg4	27,146	17.66%	20.24%	18.96%	20.24%	19.17%	20.24%
7qkpcg5	18,714	13.96%	18.26%	15.45%	21.49%	16.29%	21.49%
8qkpcg1	12,388	50.01%	50.01%	50.01%	50.01%	50.01%	50.01%
8qkpcg2	13,221	46.26%	47.89%	46.26%	47.89%	46.78%	48.40%
8qkpcg3	11,054	51.12%	53.56%	52.70%	58.03%	53.78%	58.03%
8qkpcg4	13,060	54.94%	57.07%	55.37%	57.07%	57.15%	58.61%
8qkpcg5	12,960	37.57%	42.59%	37.57%	42.59%	38.89%	42.59%
9qkpcg1	6823	63.53%	63.86%	63.53%	63.86%	63.86%	63.86%
9qkpcg2	7673	35.36%	36.45%	37.09%	37.85%	37.29%	37.85%
9qkpcg3	5291	100.51%	105.42%	102.77%	105.42%	105.11%	105.42%
9qkpcg4	8241	33.08%	36.37%	33.38%	36.37%	37.22%	42.59%
9qkpcg5	6792	56.03%	57.73%	56.10%	57.73%	56.10%	57.73%
Av.	12,012.38	27.80%	30.55%	28.74%	31.20%	29.96%	31.86%

From Tables 3 and 4, we can observe the inferiority of the GLPK solver, which is based on one of the best exact algorithms: the branch-and-cut algorithm (see [17]). GLPK realizes an average objective value of 12,012.38 compared to those reached by NSBM within less runtime. For three considered time limits, the average value of the mean objective values (V_{Mean}) reached by NSBM over five trials are better than the average value of the best objectives values computed by GLPK, where 14,924.63 for the time limit of 50 s, 15,050.58 for 100 s and 15,194.50 for 200 s. With the first limit of runtime (i.e., 50 s), if we consider only the best solutions returned by NSBM over five independent trials, we can observe that NSBM successes in improving the best solutions computed by GLPK on 44 cases and fails for only one instance. By extending the time limit to 100 seconds (resp. 200 s), the best solutions

computed by NSBM over five trials become more robust. For two last cases, NSBM dominates GLPK on all cases. Furthermore, the computational effort required by NSBM is much more interesting than that consumed by the GLPK solver.

5. Conclusions

This paper investigated a new mathematical model for designing and optimizing economic policies in an environmental protection zone. To efficiently solve the proposed problem, we propose a metaheuristic to generate and explore a series of reduced solution spaces. The proposed method embeds an efficient starting solution procedure into a sophistical neighborhood search. The staring solution procedure consists of solving the original problem by successively ignoring the constraints and the quadratic term. The neighborhood search is introduced to build and explore a series of neighborhoods from the starting solution. The performance of the proposed approach is evaluated on a group of benchmark data sets and compared with a performant integer programming solver: GLPK. The provided results show the method's success in providing high-quality solutions within reasonable runtime.

Acknowledgments: This work is partially funded by grants from the NSFC: 41501441, the Hubei Provincial Natural Science Foundation of China: 2013CFB294, and Wuhan University Grant: 2015qyw16.

Author Contributions: Xiaoliang MENG analyzed the data set; Xiaochuan SHI contributed reagents/materials/analysis tools; Lei WU conceived and designed the models and experiments; Xiaochuan SHI and Lei WU wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lee, B.; Won, D.; Park, J.H.; Kwon, L.N.; Moon, Y.H.; Kim, H.J. Patent-Enhancing Strategies by Industry in Korea Using a Data Envelopment Analysis. *Sustainability* **2016**, *8*, 901, doi:10.3390/su8090901.
2. Lee, D.; Kim, S.; Kim, S. Development of Hybrid Model for Estimating Construction Waste for Multifamily Residential Buildings Using Artificial Neural Networks and Ant Colony Optimization. *Sustainability* **2016**, *8*, 870, doi:10.3390/su8090870.
3. Li, L.; Lin, B. Green Economy Performance and Green Productivity Growth in China's Cities: Measures and Policy Implication. *Sustainability* **2016**, *8*, 947, doi:10.3390/su8090947.
4. Shim, S.O.; Park, K.B. Technology for Production Scheduling of Jobs for Open Innovation and Sustainability with Fixed Processing Property on Parallel Machines. *Sustainability* **2016**, *8*, 904, doi:10.3390/su8090904.
5. Garey, M.; Johnson, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; Freeman: New York, NY, USA, 1979.
6. Gallo, G.; Hammer, P.; Simeone, B. Quadratic knapsack problems. *Math. Program. Study* **1980**, *12*, 132–149.
7. Pisinger, D. The quadratic knapsack problem: A survey. *Discret. Appl. Math.* **2007**, *155*, 623–648.
8. Pferschy, U.; Schauer, J. The knapsack problem with conflict graphs. *J. Graph Algorithms Appl.* **2009**, *13*, 233–249.
9. Yamada, T.; Kataoka, S.; Watanabe, K. Heuristic and exact algorithms for the disjunctively constrained knapsack problem. *Inf. Process. Soc. Jpn. J.* **2002**, *43*, 2864–2870.
10. Chen, Y.; Hao, J.K. An iterated "hyperplane exploration" approach for the quadratic knapsack problem. *Comput. Oper. Res.* **2017**, *77*, 226–239.
11. Hifi, M.; Saleh, S.; Wu, L. A Fast Large Neighborhood Search for Disjunctively Constrained Knapsack Problem. In Proceedings of the Third International Symposium, ISCO 2014, Lisbon, Portugal, 5–7 March 2014; pp. 396–407.
12. Hifi, M.; Saleh, S.; Wu, L. A hybrid guided neighborhood search for the disjunctively constrained knapsack problem. *Cogent Eng.* **2015**, doi:10.1080/23311916.2015.1068969.
13. Glover, F.; Woolsey, E. Converting the 0–1 polynomial programming problem to a 0–1 linear program. *Oper. Res.* **1974**, *22*, 180–182.

14. Martello, S.; Pisinger, D.; Toth, P. Dynamic programming and strong bounds for the 0–1 knapsack problem. *Manag. Sci.* **1999**, *45*, 414–424.
15. Shaw, P. Using constraint programming and local search methods to solve vehicle routing problems. In Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming, CP98, Pisa, Italy, 26–30 October 1998; pp. 417–431.
16. Billionnet, A.; Soutif, E. An exact method based on Lagrangian decomposition for the 0–1 quadratic knapsack problem. *Eur. J. Oper. Res.* **2004**, *157*, 565–575.
17. GLPK: GNU Linear Programming Kit. Available online: https://www.gnu.org/software/glpk/ (accessed on 23 June 2012).