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 sophisticated 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., $E \subseteq \{ (i, j) \in I \times I, \ i < j \}$). 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):

\[
\begin{align*}
\text{(QP}_{QKPCG}\text{)} & \quad \text{max} \quad \sum_{i \in I} p_i x_i + \sum_{i \in I} \sum_{j > i} p_{ij} x_i x_j \\
\text{s.t.} & \quad \sum_{i \in I} w_i x_i \leq c \\
& \quad x_i + x_j \leq 1 \quad \forall (i, j) \in E \\
& \quad x_i \in \{0, 1\} \quad \forall i \in I.
\end{align*}
\]

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 \( (\text{QP}_{QKPCG}) \) can be used to establish a reasonable economic development strategy in the environmental protection zone. In \( \text{QP}_{QKPCG} \), 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 precisely, 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 \( i \)-th 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 metaheuristics, 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 \( \text{QP}_{QKPCG} \). 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 \( \text{QP}_{QKPCG} \). 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.

\[
\begin{align*}
\text{(ILP}_{QKPCG}\text{)} & \quad \max \quad \sum_{i \in I} p_i x_i + \sum_{i \in I, j \in I, j > i} p_i y_{ij} \\
& \quad \text{s.t.} \quad \sum_{i \in I} w_i x_i \leq c \\
& \quad \qquad x_i + x_j \leq 1 \quad \forall (i, j) \in E \\
& \quad \qquad x_i + x_j \leq 1 + y_{ij} \quad \forall i \in I, j \in I, j > i \\
& \quad \qquad y_{ij} \leq x_i \text{ and } y_{ij} \leq x_j \quad \forall i \in I, j \in I, j > i \\
& \quad \quad x_i \in \{0, 1\}, y_{ij} \in \{0, 1\} \quad \forall i \in I.
\end{align*}
\]

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:

\[
\begin{align*}
\text{(ILP}_{KPCG}\text{)} & \quad \max \quad \sum_{i \in I} p_i x_i \\
& \quad \text{s.t.} \quad \sum_{i \in I} w_i x_i \leq c \\
& \quad \qquad x_i + x_j \leq 1 \quad \forall (i, j) \in E \\
& \quad \quad x_i \in \{0, 1\} \quad \forall i \in I.
\end{align*}
\]

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 yielded at the first phase. Secondily, 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:

\[
\begin{align*}
\text{(ILP}_{WIS}\text{)} & \quad \text{max} \quad \sum_{i \in I} p_i x_i \\
& \quad \text{s.t.} \quad x_i + x_j \leq 1, \forall (i, j) \in E \\
& \quad \qquad x_i \in \{0, 1\}, \forall i \in I, \\
\end{align*}
\]

\[
\begin{align*}
\text{(ILP}_{K}\text{)} & \quad \text{max} \quad \sum_{i \in IS} p_i x_i \\
& \quad \text{s.t.} \quad \sum_{i \in IS} w_i x_i \leq c, \\
& \quad \qquad x_i \in \{0, 1\}, \forall i \in IS.
\end{align*}
\]

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. \hspace{1em} Set $i = \text{argmax}\{ \frac{p_i}{w_i} \mid \frac{p_k}{w_k} \geq \frac{p_i}{w_i}, k \in I \}$;
4. \hspace{1em} Add $i$ into IS: IS = IS $\cup \{i\}$;
5. \hspace{1em} 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. \hspace{1em} Set $S_{KPCG}$ as IS;
10. else
11. \hspace{1em} 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 $\alpha$ 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 $S_{QKPCG}^{\star}$, is replaced by $S_{QKPCG}$ if it has a better value. Step 4 cleans the values of $\alpha$ 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 $S_{QKPCG}^{\star}$ deduced from the starting solution.
Algorithm 2 A local search for improving a given solution

**Input:** $S_{\text{QKPCG}}$, a feasible solution of the QKPCG.

**Output:** $S^*_\text{QKPCG}$, a local optimum solution of the QKPCG.

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

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

**Input:** $S_{\text{QKPCG}}$, a starting solution of ILP$_{\text{QKPCG}}$.

**Output:** An independent set $IS$ and a reduced instance $I'$ of ILP$_{\text{QKPCG}}$.

1: Set $cnt = 0$, $I' = \emptyset$ and $IS$ to the independent set related to $S_{\text{QKPCG}}$;
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 $i = |IS| \times r$;
5: Remove $i$th item from $IS$, $i$ in $I'$ 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 $I'$ and set $cnt = cnt + 1$;
9: end if
10: end for
11: end while
12: return $IS$ and $I'$.

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$_{\text{QKPCG}}$, 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$_{\text{QKPCG}}$. The triple parameters $(\beta, r, \gamma)$ control the variety of sub solution spaces generated by Algorithm 3.
For example, if the values of $\beta$ and $r^\gamma$ 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:** $S^*_QKPCG$, 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 $S^*_QKPCG$ 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 $Ir$ associated to $S_{QKPCG}$;
   5. Apply Algorithm 1 with $Ir$ to compute a new solution $S_{QKPCG}$;
   6. Apply Algorithm 2 to determine $S_{QKPCG}$;
   7. Update $S^*_QKPCG$ if $S_{QKPCG}$ is better;
4. end while
9. return $S^*_QKPCG$.

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 1$qkpcg$–9$qkpcg$. 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 1 displays the set of values chosen for parameters in our experiment, which guarantees the best performance for NSBM on the adopted benchmark problems.

<table>
<thead>
<tr>
<th>Class</th>
<th>$n$</th>
<th>Density</th>
<th>Class</th>
<th>$n$</th>
<th>Density</th>
<th>Class</th>
<th>$n$</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>1$qkpcg$</td>
<td>100</td>
<td>2%</td>
<td>4$qkpcg$</td>
<td>150</td>
<td>2%</td>
<td>7$qkpcg$</td>
<td>200</td>
<td>2%</td>
</tr>
<tr>
<td>2$qkpcg$</td>
<td>100</td>
<td>4%</td>
<td>5$qkpcg$</td>
<td>150</td>
<td>4%</td>
<td>8$qkpcg$</td>
<td>200</td>
<td>4%</td>
</tr>
<tr>
<td>3$qkpcg$</td>
<td>100</td>
<td>8%</td>
<td>6$qkpcg$</td>
<td>150</td>
<td>8%</td>
<td>9$qkpcg$</td>
<td>200</td>
<td>8%</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Instance</th>
<th>$V_{GLPK}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1qkpcg1</td>
<td>15,482</td>
</tr>
<tr>
<td>1qkpcg2</td>
<td>11,774</td>
</tr>
<tr>
<td>1qkpcg3</td>
<td>15,129</td>
</tr>
<tr>
<td>1qkpcg4</td>
<td>17,151</td>
</tr>
<tr>
<td>1qkpcg5</td>
<td>15,147</td>
</tr>
<tr>
<td>2qkpcg1</td>
<td>9018</td>
</tr>
<tr>
<td>2qkpcg2</td>
<td>9150</td>
</tr>
<tr>
<td>2qkpcg3</td>
<td>10,002</td>
</tr>
<tr>
<td>2qkpcg4</td>
<td>12,589</td>
</tr>
<tr>
<td>2qkpcg5</td>
<td>10,810</td>
</tr>
<tr>
<td>3qkpcg1</td>
<td>5843</td>
</tr>
<tr>
<td>3qkpcg2</td>
<td>6068</td>
</tr>
<tr>
<td>3qkpcg3</td>
<td>6091</td>
</tr>
<tr>
<td>3qkpcg4</td>
<td>7978</td>
</tr>
<tr>
<td>3qkpcg5</td>
<td>6290</td>
</tr>
<tr>
<td>4qkpcg1</td>
<td>18,025</td>
</tr>
<tr>
<td>4qkpcg2</td>
<td>17,049</td>
</tr>
<tr>
<td>4qkpcg3</td>
<td>15,835</td>
</tr>
<tr>
<td>4qkpcg4</td>
<td>26,428</td>
</tr>
<tr>
<td>4qkpcg5</td>
<td>15,768</td>
</tr>
<tr>
<td>5qkpcg1</td>
<td>10,663</td>
</tr>
<tr>
<td>5qkpcg2</td>
<td>10,720</td>
</tr>
<tr>
<td>5qkpcg3</td>
<td>10,846</td>
</tr>
<tr>
<td>5qkpcg4</td>
<td>13,639</td>
</tr>
<tr>
<td>5qkpcg5</td>
<td>10,883</td>
</tr>
<tr>
<td>6qkpcg1</td>
<td>6032</td>
</tr>
<tr>
<td>6qkpcg2</td>
<td>6046</td>
</tr>
<tr>
<td>6qkpcg3</td>
<td>6429</td>
</tr>
<tr>
<td>6qkpcg4</td>
<td>7366</td>
</tr>
<tr>
<td>6qkpcg5</td>
<td>5866</td>
</tr>
<tr>
<td>7qkpcg1</td>
<td>12,589</td>
</tr>
<tr>
<td>7qkpcg2</td>
<td>10,002</td>
</tr>
<tr>
<td>7qkpcg3</td>
<td>10,720</td>
</tr>
<tr>
<td>7qkpcg4</td>
<td>10,846</td>
</tr>
<tr>
<td>7qkpcg5</td>
<td>10,883</td>
</tr>
<tr>
<td>8qkpcg1</td>
<td>13,060</td>
</tr>
<tr>
<td>8qkpcg2</td>
<td>10,002</td>
</tr>
<tr>
<td>8qkpcg3</td>
<td>10,720</td>
</tr>
<tr>
<td>8qkpcg4</td>
<td>10,846</td>
</tr>
<tr>
<td>8qkpcg5</td>
<td>10,883</td>
</tr>
<tr>
<td>9qkpcg1</td>
<td>15,482</td>
</tr>
<tr>
<td>9qkpcg2</td>
<td>11,774</td>
</tr>
<tr>
<td>9qkpcg3</td>
<td>15,129</td>
</tr>
<tr>
<td>9qkpcg4</td>
<td>17,151</td>
</tr>
<tr>
<td>9qkpcg5</td>
<td>15,147</td>
</tr>
<tr>
<td>10qkpcg1</td>
<td>8,910</td>
</tr>
<tr>
<td>10qkpcg2</td>
<td>9,150</td>
</tr>
<tr>
<td>10qkpcg3</td>
<td>9,091</td>
</tr>
<tr>
<td>10qkpcg4</td>
<td>8,930</td>
</tr>
<tr>
<td>10qkpcg5</td>
<td>8,970</td>
</tr>
<tr>
<td>11qkpcg1</td>
<td>9,002</td>
</tr>
<tr>
<td>11qkpcg2</td>
<td>9,144</td>
</tr>
<tr>
<td>11qkpcg3</td>
<td>9,285</td>
</tr>
<tr>
<td>11qkpcg4</td>
<td>9,426</td>
</tr>
<tr>
<td>11qkpcg5</td>
<td>9,567</td>
</tr>
<tr>
<td>12qkpcg1</td>
<td>9,091</td>
</tr>
<tr>
<td>12qkpcg2</td>
<td>9,144</td>
</tr>
<tr>
<td>12qkpcg3</td>
<td>9,285</td>
</tr>
<tr>
<td>12qkpcg4</td>
<td>9,426</td>
</tr>
<tr>
<td>12qkpcg5</td>
<td>9,567</td>
</tr>
</tbody>
</table>

Table 2. Parameters setting used to perform NSBM.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\gamma$</th>
<th>NSBM time in Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>values</td>
<td>20</td>
<td>15</td>
<td>20</td>
<td>(50,100,200)</td>
</tr>
</tbody>
</table>

Table 3. Performance of NSBM vs GLPK on the benchmark instances.
Table 4. Percentage improvement of NSBM vs GLPK on the benchmark instances.

<table>
<thead>
<tr>
<th>Instance</th>
<th>V_{GLPK}</th>
<th>t = 50</th>
<th>NSBM</th>
<th>t = 100</th>
<th>NSBM</th>
<th>t = 200</th>
<th>NSBM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>IMP_Mean</td>
<td>IMP_Best</td>
<td>IMP_Mean</td>
<td>IMP_Best</td>
<td>IMP_Mean</td>
<td>IMP_Best</td>
</tr>
<tr>
<td>1qkpcg1</td>
<td>15,482</td>
<td>2.98%</td>
<td>5.12%</td>
<td>3.93%</td>
<td>5.53%</td>
<td>5.16%</td>
<td>7.49%</td>
</tr>
<tr>
<td>1qkpcg2</td>
<td>11,774</td>
<td>−1.21%</td>
<td>1.28%</td>
<td>0.75%</td>
<td>3.77%</td>
<td>1.97%</td>
<td>4.23%</td>
</tr>
<tr>
<td>1qkpcg3</td>
<td>15,129</td>
<td>1.12%</td>
<td>3.63%</td>
<td>1.21%</td>
<td>3.63%</td>
<td>3.20%</td>
<td>4.72%</td>
</tr>
<tr>
<td>1qkpcg4</td>
<td>17,151</td>
<td>14.04%</td>
<td>14.41%</td>
<td>14.15%</td>
<td>14.41%</td>
<td>14.25%</td>
<td>14.62%</td>
</tr>
<tr>
<td>1qkpcg5</td>
<td>15,147</td>
<td>0.46%</td>
<td>2.32%</td>
<td>1.19%</td>
<td>2.32%</td>
<td>1.53%</td>
<td>2.32%</td>
</tr>
<tr>
<td>2qkpcg1</td>
<td>9,018</td>
<td>17.55%</td>
<td>20.66%</td>
<td>19.78%</td>
<td>24.25%</td>
<td>20.81%</td>
<td>24.25%</td>
</tr>
<tr>
<td>2qkpcg2</td>
<td>10,002</td>
<td>11.78%</td>
<td>14.07%</td>
<td>12.71%</td>
<td>15.46%</td>
<td>14.08%</td>
<td>15.46%</td>
</tr>
<tr>
<td>2qkpcg3</td>
<td>12,589</td>
<td>22.55%</td>
<td>23.91%</td>
<td>22.55%</td>
<td>23.91%</td>
<td>23.91%</td>
<td>23.91%</td>
</tr>
<tr>
<td>2qkpcg4</td>
<td>10,810</td>
<td>15.90%</td>
<td>16.37%</td>
<td>16.02%</td>
<td>16.37%</td>
<td>16.02%</td>
<td>16.37%</td>
</tr>
<tr>
<td>3qkpcg1</td>
<td>5,843</td>
<td>22.77%</td>
<td>27.59%</td>
<td>22.77%</td>
<td>27.59%</td>
<td>25.57%</td>
<td>27.59%</td>
</tr>
<tr>
<td>3qkpcg2</td>
<td>6,068</td>
<td>17.83%</td>
<td>21.01%</td>
<td>19.45%</td>
<td>21.01%</td>
<td>21.01%</td>
<td>21.01%</td>
</tr>
<tr>
<td>3qkpcg3</td>
<td>6,040</td>
<td>17.94%</td>
<td>19.16%</td>
<td>18.22%</td>
<td>19.16%</td>
<td>18.22%</td>
<td>19.16%</td>
</tr>
<tr>
<td>3qkpcg4</td>
<td>7,978</td>
<td>0.11%</td>
<td>0.11%</td>
<td>0.13%</td>
<td>0.16%</td>
<td>0.13%</td>
<td>0.16%</td>
</tr>
<tr>
<td>3qkpcg5</td>
<td>6,290</td>
<td>13.77%</td>
<td>16.85%</td>
<td>13.77%</td>
<td>16.85%</td>
<td>13.77%</td>
<td>16.85%</td>
</tr>
<tr>
<td>4qkpcg1</td>
<td>18,025</td>
<td>3.33%</td>
<td>7.94%</td>
<td>3.53%</td>
<td>7.94%</td>
<td>5.03%</td>
<td>7.94%</td>
</tr>
<tr>
<td>4qkpcg2</td>
<td>17,049</td>
<td>6.02%</td>
<td>9.70%</td>
<td>8.39%</td>
<td>12.85%</td>
<td>9.97%</td>
<td>12.85%</td>
</tr>
<tr>
<td>4qkpcg3</td>
<td>15,125</td>
<td>26.41%</td>
<td>29.81%</td>
<td>28.99%</td>
<td>31.76%</td>
<td>29.70%</td>
<td>31.76%</td>
</tr>
<tr>
<td>4qkpcg4</td>
<td>26,428</td>
<td>14.04%</td>
<td>14.41%</td>
<td>14.15%</td>
<td>14.41%</td>
<td>14.25%</td>
<td>14.62%</td>
</tr>
<tr>
<td>4qkpcg5</td>
<td>15,147</td>
<td>0.46%</td>
<td>2.32%</td>
<td>1.19%</td>
<td>2.32%</td>
<td>1.53%</td>
<td>2.32%</td>
</tr>
<tr>
<td>5qkpcg1</td>
<td>10,663</td>
<td>37.09%</td>
<td>39.27%</td>
<td>37.16%</td>
<td>39.27%</td>
<td>38.28%</td>
<td>39.46%</td>
</tr>
<tr>
<td>5qkpcg2</td>
<td>10,720</td>
<td>37.50%</td>
<td>40.56%</td>
<td>37.16%</td>
<td>40.56%</td>
<td>39.52%</td>
<td>40.56%</td>
</tr>
<tr>
<td>5qkpcg3</td>
<td>10,846</td>
<td>35.59%</td>
<td>36.69%</td>
<td>37.31%</td>
<td>37.74%</td>
<td>37.80%</td>
<td>39.52%</td>
</tr>
<tr>
<td>5qkpcg4</td>
<td>13,639</td>
<td>32.18%</td>
<td>34.76%</td>
<td>31.42%</td>
<td>32.34%</td>
<td>32.28%</td>
<td>32.34%</td>
</tr>
<tr>
<td>5qkpcg5</td>
<td>10,883</td>
<td>40.27%</td>
<td>43.41%</td>
<td>40.52%</td>
<td>43.41%</td>
<td>40.71%</td>
<td>43.41%</td>
</tr>
<tr>
<td>6qkpcg1</td>
<td>6,032</td>
<td>44.21%</td>
<td>44.23%</td>
<td>45.37%</td>
<td>48.69%</td>
<td>46.55%</td>
<td>48.69%</td>
</tr>
<tr>
<td>6qkpcg2</td>
<td>6,041</td>
<td>49.87%</td>
<td>52.18%</td>
<td>52.18%</td>
<td>52.18%</td>
<td>50.62%</td>
<td>52.18%</td>
</tr>
<tr>
<td>6qkpcg3</td>
<td>5,866</td>
<td>52.99%</td>
<td>57.36%</td>
<td>52.99%</td>
<td>57.36%</td>
<td>52.99%</td>
<td>57.36%</td>
</tr>
<tr>
<td>6qkpcg4</td>
<td>27,166</td>
<td>17.66%</td>
<td>20.24%</td>
<td>18.96%</td>
<td>20.24%</td>
<td>19.17%</td>
<td>20.24%</td>
</tr>
<tr>
<td>6qkpcg5</td>
<td>18,714</td>
<td>13.96%</td>
<td>18.26%</td>
<td>15.45%</td>
<td>18.26%</td>
<td>16.29%</td>
<td>18.26%</td>
</tr>
<tr>
<td>7qkpcg1</td>
<td>12,388</td>
<td>50.01%</td>
<td>50.01%</td>
<td>50.01%</td>
<td>50.01%</td>
<td>50.01%</td>
<td>50.01%</td>
</tr>
<tr>
<td>7qkpcg2</td>
<td>13,221</td>
<td>46.26%</td>
<td>47.89%</td>
<td>46.26%</td>
<td>47.89%</td>
<td>46.78%</td>
<td>48.40%</td>
</tr>
<tr>
<td>7qkpcg3</td>
<td>11,054</td>
<td>51.12%</td>
<td>53.56%</td>
<td>52.70%</td>
<td>58.03%</td>
<td>53.78%</td>
<td>58.03%</td>
</tr>
<tr>
<td>7qkpcg4</td>
<td>8,241</td>
<td>33.08%</td>
<td>36.37%</td>
<td>33.38%</td>
<td>36.37%</td>
<td>33.22%</td>
<td>36.59%</td>
</tr>
<tr>
<td>7qkpcg5</td>
<td>6,792</td>
<td>56.03%</td>
<td>57.73%</td>
<td>56.10%</td>
<td>57.73%</td>
<td>56.10%</td>
<td>57.73%</td>
</tr>
<tr>
<td>Av.</td>
<td>12,012.38</td>
<td>27.80%</td>
<td>30.55%</td>
<td>28.74%</td>
<td>31.20%</td>
<td>29.96%</td>
<td>31.86%</td>
</tr>
</tbody>
</table>

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 sophisticated 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: 2015gyw16.

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


© 2017 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).