algorithms

MDPI

Article

The Knapsack Problem with Conflict Pair Constraints on Bipartite Graphs and Extensions

Abraham P. Punnen and Jasdeep Dhahan *

Department of Mathematics, Simon Fraser University, Surrey, BC V5A 1S6, Canada; apunnen@sfu.ca
* Correspondence: jasdeep_dhahan@sfu.ca

Abstract: In this paper, we study the knapsack problem with conflict pair constraints. After a thorough literature survey on the topic, our study focuses on the special case of bipartite conflict graphs. For complete bipartite (multipartite) conflict graphs, the problem is shown to be NP-hard but solvable in pseudo-polynomial time, and it admits an FPTAS. Extensions of these results to more general classes of graphs are also presented. Further, a class of integer programming models for the general knapsack problem with conflict pair constraints is presented, which generalizes and unifies the existing formulations. The strength of the LP relaxations of these formulations is analyzed, and we discuss different ways to tighten them. Experimental comparisons of these models are also presented to assess their relative strengths. This analysis disclosed various strong and weak points of different formulations of the problem and their relationships to different types of problem data. This information can be used in designing special purpose algorithms for KPCC involving a learning component.

Keywords: knapsack problem; stable sets; conflict pair constraints; combinatorial optimization; algorithms

Citation: Punnen, A.P.; Dhahan, J. The Knapsack Problem with Conflict Pair Constraints on Bipartite Graphs and Extensions. *Algorithms* **2024**, *17*, 219. https://doi.org/10.3390/a17050219

Academic Editor: Frank Werner

Received: 24 February 2024
Revised: 7 May 2024
Accepted: 9 May 2024
Published: 18 May 2024

1. Introduction

Let $G = (V, E)$ be an undirected graph, and for each $i \in V = \{1, 2, \ldots, n\}$ an ordered pair (c_i, a_i) is prescribed such that $a_i, c_i \geq 0$ for all i. This non-negativity assumption can be relaxed in some of the results discussed in this paper. We refer to c_i as the *cost* of i and a_i as the *weight* of i. Let $b \in \mathbb{R}^{=} \cup \{0\}$ be a given budget. For any $S \subseteq V$, let $c(S) = \sum_{i \in S} c_i$ be its cost and $a(S) = \sum_{i \in S} a_i$ be its weight. Then, the *knapsack problem with conflict pair constraints* (KPCC) is to find a stable set S in G satisfying $a(S) \leq b$ such that $c(S)$ is as large as possible. The KPCC can be formulated as a 0-1 programming problem as follows:

$$\begin{aligned} \text{Maximize} \quad & \mathbf{cx} \\ \text{Subject to:} \quad & \mathbf{ax} \leq b \\ & x_i + x_j \leq 1 \text{ for all } (i,j) \in E \\ & \mathbf{x} \in \{0,1\}^n \end{aligned}$$

where $\mathbf{c} = (c_1, c_2, \ldots, c_n)$, $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{x}^T = (x_1, x_2, \ldots, x_n)$.

The KPCC has been investigated by many researchers, and it is known under different names. Some authors call it the *maximum independent set problem with a budget constraint* [1–3], while some others call it the *disjunctively constrained knapsack problem* [4–10]. We use the name knapsack problem with conflict pair constraints since it belongs to the broader class of combinatorial optimization problems with conflict pair constraints [11–20].

Bandyapadhyay [1] notes that the maximum weight stable set problem (MWSS) has many applications in various fields, including scheduling, wireless networks, computer graphics, map labeling, and molecular biology. The budget restriction has a relevant interpretation in most of these applications, and it provides motivation to study KPCC. Moreover, specific applications of KPCC include job scheduling in computers and selecting

a non-interfering set of transmitters [1]. Other applications of KPCC also have been discussed in the literature. Further, KPCC is used in the Dantzig–Wolfe decomposition formulation of the two-dimensional bin packing problem [21].

2. Literature Review

Combinatorial optimization problems with conflict pair constraints have recently received considerable attention from the research community. These include conflict pair versions of the minimum spanning tree problem [11–14,19,20], shortest path problem [14], minimum cost matching problem [14,17], minimum cost bin packing [22,23], and maximum flow problem [18]. KPCC is yet another member in this class of optimization problems. Conflict pair constraints can easily be absorbed into a quadratic objective function. Thus, these types of problems can be formulated as quadratic combinatorial optimization problems. However, the special structure of the conflict pair constraints makes the problem interesting to study independently.

The multiple choice knapsack problem studied by Chandra et al. [24] and Nauss [25] in 1975 is essentially a KPCC, where the underlying graph is a collection of disjoint cliques. The continuous version of this problem was investigated by Ibaraki et al. [26]. Another special case of KPCC, where $c_j = 1$ for all j, was mentioned by Bar-Noy et al. [27] in the context of resource allocation. The general problem KPCC was introduced by Yamada, Kataoka, and Watanab [10], and they proposed exact and heuristic algorithms to solve the problem. Hifi and Michrafy [6] presented different exact algorithms based on MIP formulations. Bettinelli, Cacchiani, and Malaguti [28] proposed yet another class of the branch and bound algorithm, along with extensive experimental analysis. Moreover, they presented comparisons of MIP formulations using general purpose solvers. Salem et al. [29] investigated the convex hull of feasible solutions of KPCC, i.e., the KPCC polytope. The authors obtained necessary and sufficient conditions for odd cycle inequalities associated with the stable set problem to be facet-defining for the KPCC polytope. Similar conditions were developed for the cover inequalities of the knapsack polytope in connection with the KPCC polytope. They also analyzed different lifting procedures to obtain additional valid inequalities. Exploiting these inequalities, the authors proposed an effective branch and bound algorithm to solve KPCC and reported the results of extensive computational experiments.

Heuristic algorithms for KPCC have been investigated by many researchers, and the results of extensive experimental analysis with these algorithms are available. Hifi and Michrafy [5] developed a reactive-local-search-based algorithm and also generated a class of test instances based on the ideas from [10]. Akeb et al. [4] proposed local-branching-based heuristics and provided an experimental analysis based on instances generated by [5]. Hifi and Otmani [7] proposed a scatter-search-based metaheuristic algorithm along with results of an experimental analysis using test instances from [5]. Recently, Hifi [9] proposed an iterative-rounding-search-based algorithm. Their experimental results using test instances from [5] were promising and were able to obtain the best known solutions for many benchmark instances. Salem et al. [30] developed a probabilistic tabu search algorithm to solve KPCC and reported extensive computational results.

Pferschy and Schauer [31] investigated KPCC for several special cases of the graph G. They showed that KPCC can be solved in pseudo-polynomial time when G is a tree and more generally when G is a graph with bounded treewidth or chordal. Fully polynomial-time approximation schemes (FPTASs) are also derived for these classes of graphs. Further, they proved that KPCC remains strongly NP-hard on perfect graphs. It may be noted that the stable set problem is polynomial-time-solvable on perfect graphs. Bandyapadhyay [1] also considered KPCC on trees, but the algorithm discussed in [31] is faster for this case. Moreover, [1] presents pseudopolynomial algorithms for cycles, forests, k-outerplanar graphs, and interval graphs. In another paper, Pferschy and Schauer [32] investigated approximation algorithms for KPCC under various restrictions on G. They showed that KPCC admits FPTASs on weakly chordal graphs and is solvable by PTAS (not FPTASs) on planar graphs.

Kalra et al. [2] studied another special case of KPCC, where $a_j = 1$ for all $j = 1, 2, \ldots, n$ and G is a multipartite graph. They proposed a $\frac{1}{k}$-approximation algorithm, where k is the number of partite sets, and proved that the performance bound established is tight. If $a_j = 1$ and c_j equal to the degree of j, for $j = 1, 2, \ldots, n$, KPCC reduces to the maximum independent vertex coverage problem (MIVC). The authors proved approximation hardness results for MIVC and showed that MIVC is NP-hard on bipartite graphs. Further, they established bounds on the integrality for the LP relaxation of a clique-based formulation of MVIP. Bandyapadhyay [1] investigated another special case of KPCC, where $c_j = 1$ for all j. This problem is called the maximum budget independent set problem (MBIS), and the author presented a factor d polynomial time approximation algorithm for the problem on $(d + 1)$-claw free graphs.

In KPCC, if we restrict the budget constraint to be satisfied as an equality, we obtain an instance of the *exact knapsack problem with conflict pair constraints* (KPCC-exact). The feasibility version of this problem was investigated by Milanic and Monnot [3,33], where it was shown that the problem is strongly NP-complete on bipartite graphs with maximum vertex degree of three. The authors also presented a pseudo-polynomial algorithm for the problem in cographs, chordal graphs, and interval graphs.

A generalization of KPCC involving more than one knapsack constraint was studied by Gabriel [34], where a Dantzig–Wolfe decomposition scheme was proposed. Atamtürk et al. [35] used conflict graphs in developing algorithms for 0-1integer programs.

The *quadratic knapsack problem* (QKP) [36] can be stated as

$$\begin{aligned} \text{Maximize} \quad & \mathbf{x}^t Q \mathbf{x} + \mathbf{c}\mathbf{x} \\ \text{Subject to:} \quad & \mathbf{a}\mathbf{x} \leq b \\ & \mathbf{x} \in \{0, 1\}^n \end{aligned}$$

where $Q = (q_{ij})$ is an $n \times n$ matrix.

Although the QKP appears to be more general than KPCC, we now observe that QKP and KPCC are essentially equivalent, in the sense that one is a special case of the other.

Theorem 1. *QKP and KPCC are equivalent.*

Proof. Define the matrix $Q = (q_{ij})$ such that $q_{ij} = -M$, where M is a large positive number if $(i, j) \in E$ and $q_{ij} = 0$ otherwise. With this choice of Q, the problem KPCC reduces to QKP. Thus, KPCC is a special case of QKP.

Let us now prove the converse. That is, QKP is a special case of KPCC. This is, however, achieved by appropriate modifications in the well-known reduction of a quadratic unconstrained binary optimization problem (QUBO) to the maximum weight stable set problem [37,38] to handle the budget constraint. Thus, KPCC can be viewed as a generalization of QKP. □

For further details on QUBO, refer to [39]. The major contributions of the paper can be summarized as follows.

- We present new complexity results on KPCC along with a thorough state-of-the-art review of the literature. Also, we show that KPCC and QKP are equivalent.
- When the conflict graph is a complete bipartite graph, it is shown that KPCC decomposes into two knapsack problems. As a consequence, we have new polynomially solvable (and pseudo-polynomially solvable) special cases of KPCC and have special cases of KPCC that admit FPTASs. These results are extended to more general classes of graphs that accept a blakbox oracle with special properties. On a star (which is a special bipartite graph), the problem is shown to be NP-hard.
- A new integer programming formulation that unifies and generalizes existing formulations of KPCC on general graphs is given. The strengths of the LP relaxation of this formulation and of special cases are analyzed theoretically.

- Different methods are proposed to tighten our general integer programming formulation, and these methods are compared using the general purpose solver Gurobi.

3. KPCC on Bipartite Graphs

Let $G = (V_1 \cup V_2, E)$ be a bipartite graph with the generic bipartition of its vertex set as $V_1 \cup V_2$, where $V_1 = \{1, 2, \ldots, m\}$ and $V_2 = \{m+1, m+2, \ldots, m+n\}$. Note that KPCC is trivial to solve when G is a complete graph and is the same as the knapsack problem when G is a collection of isolated nodes. When G is a tree, Pferschy and Schauer [31] showed that KPCC is NP-hard but solvable in $O(nb^2)$ time and $O(nb)$ space. Further, they showed that the algorithm can be modified to obtain a fully polynomial approximation scheme (FPTASs) for KPCC. Trees, in some sense, can be viewed as an extreme case of sparse connected bipartite graphs. Let us now look at KPCC on the complete bipartite graph $K_{m,n} = (V_1 \cup V_2, E)$. Although KPCC is trivial on a complete graph, this simplicity does not extend to $K_{m,n}$.

Theorem 2. *KPCC is NP-hard even if the conflict graph is a star (i.e., $K_{1,n}$). Further, on a complete bipartite graph, KPCC can be solved in pseudopolynomial time and it admits FPTASs.*

Proof. It is not difficult to reduce the knapsack problem to a KPCC on $K_{1,n}$ and this establishes the first part of the theorem. To prove the second part, note that for any solution $\mathbf{x}$ of a KPCC on $K_{m,n}$, if $x_i = 1$ for any $i \in V_1$ then by the conflict pair constraints, $x_i = 0$ for all $i \in V_2$ and vice versa. Thus, KPCC on $K_{m,n}$ reduces to two knapsack problems

$$\begin{aligned} \text{KP1: Maximize} \quad & \sum_{j \in V_1} c_j x_j \\ \text{Subject to:} \quad & \sum_{j \in V_1} a_j x_j \leq b \\ & x_j \in \{0,1\} \text{ for all } j \in V_1 \end{aligned}$$

and

$$\begin{aligned} \text{KP2: Maximize} \quad & \sum_{j \in V_2} c_j x_j \\ \text{Subject to:} \quad & \sum_{j \in V_2} a_j x_j \leq b \\ & x_j \in \{0,1\} \text{ for all } j \in V_2 \end{aligned}$$

The best solution amongst the solutions of KP1 and KP2 solves KPCC on $K_{m,n}$. Since the knapsack problem can be solved in pseudopolynomial time and it admits FPTASs, the second part of the theorem follows. □

From the above theorem, if KP1 and KP2 are solvable in polynomial time, then the corresponding KPCC on $K_{m,n}$ is also solvable in polynomial time. If $a_j = 1$ for all $j \in V_1 \cup V_2$, then obviously KP1 and KP2 are solvable in polynomial time; hence, the corresponding KPCC on $K_{m,n}$ is also solvable in polynomial time. This is interesting since KPCC on a general bipartite graph is NP-hard even if $a_j = 1$ for all j [31]. Deineko and Woeginger [40] showed that the cross-ratio-ordered bounded knapsack problem on n variables can be solved in $O(n)$ time. This, together with Theorem 2, leads to the following theorem.

Theorem 3. *If the costs c_i and weight a_i for $i \in V_1 \cup V_2$ satisfy $\frac{a_{i+1}}{a_i} \leq \left\lfloor \frac{c_{i+1}}{c_i} \right\rfloor$ for $i = 1, 2, \ldots m-1$ and $\frac{a_{i+1}}{a_i} \leq \left\lfloor \frac{c_{i+1}}{c_i} \right\rfloor$ for $i = m+1, m+2, \ldots m+n-1$, then KPCC on $K_{m,n}$ can be solved in polynomial time.*

We can also mix conditions for different solvable cases for KP1 and KP2 to obtain solvable instances of KPCC on $K_{m,n}$. For example, if $a_i = \alpha$ for all $i \in V_1$ and $\frac{a_{i+1}}{a_i} \leq \left\lfloor \frac{c_{i+1}}{c_i} \right\rfloor$ for $i = m+1, m+2, \ldots m+n-1$, then KPCC on $K_{m,n}$ can also be solved in polynomial time. Later, we present additional conditions for polynomially solvable special cases of the knapsack problem that can be used in different combinations for KP1 and KP2 to produce polynomially solvable cases of KPCC on $K_{m,n}$.

Let $G = (V_1 \cup V_2, E)$ be an undirected graph such that the subgraphs induced by V_1 and V_2 belong to a family $\mathcal{F}$ of graphs with specific properties. We call such graphs $\mathcal{F}$-bipartite (see Figure 1 for an example). If an $\mathcal{F}$-bipartite graph contain all edges (i, j) for $i \in V_1, j \in V_2$, then it is called a complete $\mathcal{F}$-bipartite graph. Bonamy et al. [41] studied $\mathcal{F}$-bipartite graphs, where the induced subgraph of V_1 is a collection of isolated nodes and the induced subgraph of V_2 is a k-degenerate graph. Depending on the properties imposed on the family $\mathcal{F}$, the $\mathcal{F}$-bipartite structure can have algorithmic advantages in solving some optimization problems. For example, $\mathcal{F}$ could contain all classes of graphs on which the KPCC can be solved in pseudo-polynomial time. In this case, $\mathcal{F}$ includes graphs that are isolated vertices, trees, complete bipartite graphs, complete graphs, chordal graphs, interval graphs, cographs, and circular arc graphs.

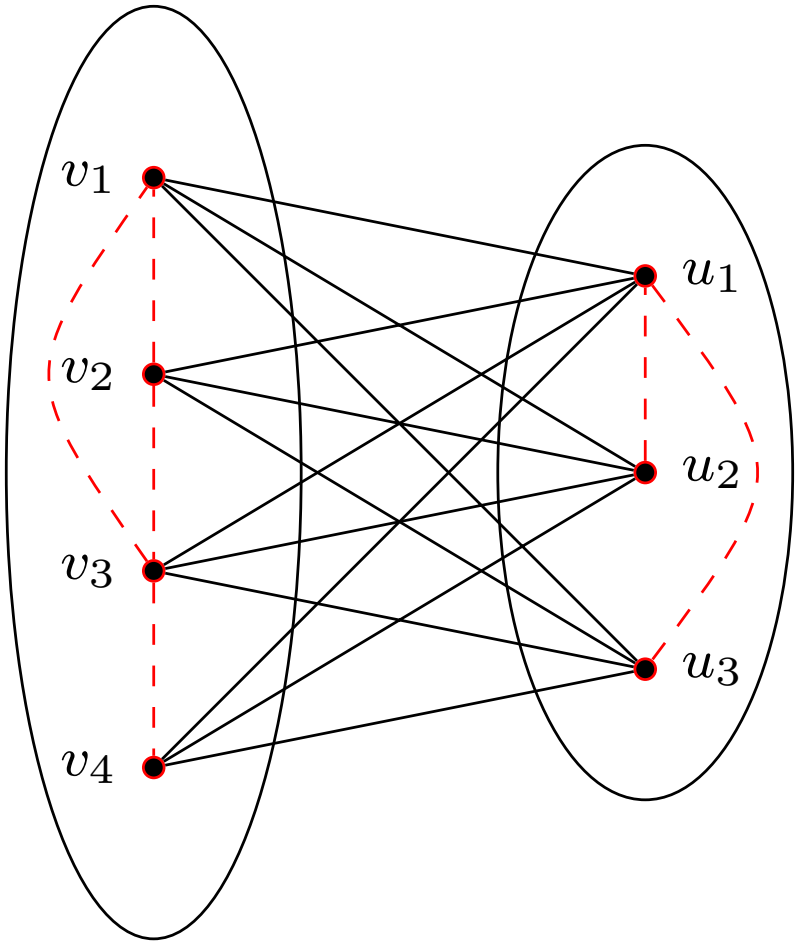

Figure 1. A complete $\mathcal{F}$-bipartite graph with $V_1 = \{v_1, v_2, v_3, v_4\}$ and $V_2 = \{u_1, u_2, u_3\}$. The subgraph induced by V_1 is a cycle and a pendant vertex, and the subgraph induced by V_2 is tree (path).

Assume that for a $\mathcal{F}$-bipartite graph, the bipartition V_1 and V_2 along with the structure of the subgraphs induced by V_1 and V_2 are given. Then, the results obtained in the previous sections on the solvability of KPCC can be extended to $\mathcal{F}$-bipartite graphs.

Theorem 4. *KPCC can be solved in pseudo-polynomial time on a complete $\mathcal{F}$-bipartite graph $G = (V_1 \cup V_2, E)$ if KPCC can be solved in pseudo-polynomial time on graphs in $\mathcal{F}$. Further, KPCC admits FPTASs on G whenever KPCC admits FPTASs on graphs in $\mathcal{F}$.*

Proof. The proof of the above theorem is similar to that of Theorem 2, except that instead of KP1 and KP2 we will have problems of the type KPCC but with reduced sizes. However, we present the complete proof of the theorem here to facilitate further related discussions. Let $G_1 = (V_1, E_1)$ be the subgraph of G induced by V_1 and $G_2 = (V_2, E_2)$ be the subgraph of G induced by V_2. Since every vertex in V_1 is connected to every vertex in V_2, for any solution $\mathbf{x}$ of the KPCC, if $x_i = 1$ for any $i \in V_1$ then $x_i = 0$ for all $i \in V_2$ and vice versa. Thus, KPCC reduces to the following reduced KPCCs.

$$\begin{aligned} \text{KPCC1: Maximize} \quad & \sum_{j \in V_1} c_j x_j \\ \text{Subject to:} \quad & \sum_{j \in V_1} a_j x_j \leq b \\ & x_i + x_j \leq 1 \text{ for } (i,j) \in E_1 \\ & x_j \in \{0,1\} \text{ for all } j \in V_1 \end{aligned}$$

and

$$\begin{aligned} \text{KPCC2: Maximize} \quad & \sum_{j \in V_2} c_j x_j \\ \text{Subject to:} \quad & \sum_{j \in V_2} a_j x_j \leq b \\ & x_i + x_j \leq 1 \text{ for } (i,j) \in E_2 \\ & x_j \in \{0,1\} \text{ for all } j \in V_2 \end{aligned}$$

The best solution amongst the solutions of KPCC1 and KPCC2 provides an optimal solution to KPCC. By assuming the properties of $\mathcal{F}$, KPCC1 and KPCC2 can be solved in pseudo-polynomial time. The proof of the second part of the theorem also follows in an analogous way. □

Theorem 4 is a proper generalization of Theorem 2.

The results discussed above extend in a natural way to complete multi-partite graphs. However, to take any computational advantage we need to know the partite sets a priori or need to compute them. Bipartite graphs can be recognized in polynomial time but recognizing p-partite graphs for $p > 2$ is NP-complete [42]. If there are p partite sets, then we will be solving p knapsack problems.

General Bipartite Graphs

We now consider KPCC on a general bipartite graph $G = (V_1 \cup V_2, E)$, where $V_1 = \{1, 2, \ldots, m\}$ and $V_2 = \{m+1, m+2, \ldots, m+n\}$. Note that we are considering a maximization problem. Consider the maximization version of a combinatorial optimization problem such that the objective function value of its feasible solutions are positive. Let $\mathbf{x}^*$ be any feasible solution to such a problem and $\mathbf{x}^0$ be a corresponding optimal solution. Then, $\mathbf{x}^*$ is said to be ϵ-optimal if $\frac{OBJ(\mathbf{x}^0)}{OBJ(\mathbf{x}^*)} \leq \epsilon$, where $OBJ(\mathbf{x})$ is the objective function value of the solution $\mathbf{x}$. When $\epsilon = 1$, then $\mathbf{x}^*$ is an optimal solution. Consider KP1 and KP2, as indicated in Theorem 2. Let $\bar{\mathbf{x}}_1$ and $\bar{\mathbf{x}}_2$ be ϵ-optimal solutions to KP1 and KP2, respectively. Define the solutions $\hat{\mathbf{x}}^1$ and $\hat{\mathbf{x}}^2$, where

$$\hat{x}_j^1 = \begin{cases} \bar{x}_j^1 & \text{if æ} \in V_1 \\ 0 & \text{otherwise} \end{cases} \text{ and } \hat{x}_j^2 = \begin{cases} \bar{x}_j^2 & \text{if æ} \in V_2 \\ 0 & \text{otherwise} \end{cases}$$

Theorem 5. *The best solution amongst* $\hat{\mathbf{x}}^1$ *and* $\hat{\mathbf{x}}^2$ *is a* 2ϵ*-optimal solution to KPCC on* $G = (V_1 \cup V_2, E)$.

Proof. Let $\mathbf{x}^0$ be an optimal solution to KPCC on $G = (V_1 \cup V_2, E)$. Define the solutions $\mathbf{x}^*$ and $\mathbf{x}^{**}$ such that

$$x_j^* = \begin{cases} x_j^0 & \text{if æ} \in V_1 \\ 0 & \text{otherwise} \end{cases} \text{ and } x_j^{**} = \begin{cases} x_j^0 & \text{if æ} \in V_2 \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathbf{x}^{1*}$ be the restriction of $\mathbf{x}^*$ to V_1 and $\mathbf{x}^{2**}$ be the restriction of $\mathbf{x}^{**}$ to V_2. Then, $\mathbf{x}^{1*}$ and $\mathbf{x}^{2**}$ are feasible solutions of KP1 and KP2, respectively. Then, by the ϵ-optimality of $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ to KP1 and KP2, we have

$$\sum_{j \in V_1} c_j \hat{x}_j^1 \leq \epsilon \left(\sum_{j \in V_1} c_j x_j^{1*} \right) = \epsilon \mathbf{c}\mathbf{x}^* \tag{1}$$

and

$$\sum_{j \in V_2} c_j \hat{x}_j^2 \leq \epsilon \left(\sum_{j \in V_2} c_j x_j^{2**} \right) = \epsilon \mathbf{c}\mathbf{x}^{**} \tag{2}$$

Without a loss of generality, assume $\mathbf{c}\hat{\mathbf{x}}^1 \geq \mathbf{c}\hat{\mathbf{x}}^2$. Then,

$$\mathbf{c}\mathbf{x}^0 = \mathbf{c}\mathbf{x}^* + \mathbf{c}\mathbf{x}^{**} \leq \epsilon \mathbf{c}\hat{\mathbf{x}}^1 + \epsilon \mathbf{c}\hat{\mathbf{x}}^2 \leq 2\epsilon \mathbf{c}\hat{\mathbf{x}}^1.$$

□

An approximation algorithm $\mathcal{A}$ for a combinatorial optimization problem (in maximization form) is said to be a *δ-fully polynomial time approximation scheme* (δ-FPTASs) if it guarantees a solution $\mathbf{x}^*$ such that $OBJ(X^0) \leq \delta(1+\epsilon)OBJ(\mathbf{x}^*)$, where $\mathbf{x}^0$ is an optimal solution and $\mathcal{A}$ runs in polynomial time in the input size and $\frac{1}{\epsilon}$ for any $\epsilon > 0$. Thus, a $1 - FPTASs$ is precisely an $FPTASs$. Since KP1 and KP2 admits FPTASs, from Theorem 5, KPCC admits $2 - FPTASs$. This result is interesting because the problem is strongly NP-complete on bipartite graphs with maximum vertex degree of three [3,33]. Further, when KP1 and KP2 are solvable in polynomial time, KPCC on a general bipartite graph can be solved by a polynomial time 2-approximation algorithm. When $a_j = 1$ for all j, Theorem 5 reduces to a corresponding result obtained by Kalra et al. [2], and for the special case, they showed that the performance bound is asymptotically tight. We summarize these observations in the following corollary.

Corollary 1. *KPCC on a bipartite graph admits* 2*-FPTASs. Further, when KP1 and KP2 are solvable in polynomial time, then a* 2*-optimal solution to KPCC on a bipartite graph can be obtained in polynomial time, and in this case the bound 2 is asymptotically tight.*

The concept of $\mathcal{F}$-bipartite graphs can be extended to the more general case of $\mathcal{F}$-multipartite graphs. Let $G = \left(\cup_{k=1}^{p} V_k, E \right)$ be an undirected graph such that the subgraphs induced by V_k for any $k = 1, 2, \ldots, p$ belong to the family $\mathcal{F}$. We call G a $\mathcal{F}$-multipartite graph. If $(i, j) \in E$ for any $i \in V_r, j \in V_s$ for $r \neq s, r, s = 1, 2, \ldots p$, then G is called a complete $\mathcal{F}$-multipartite graph. Theorem 4 and the discussions that follow extend in a natural way to complete $\mathcal{F}$-multipartite graphs.

4. Integer Programming Formulations

Let us now discuss some integer programming formulations of the KPCC. For each vertex $i \in V$, let $A(i) = \{j \in V : (i, j) \in E\}$ and the sets $N_i(1), N_i(2), \ldots, N_i(p_i)$ be a given partition

of $A(i)$, where $N_i(k) \neq \emptyset$ for any $k = 1, 2, \ldots p_i$, $N_i(r) \cap N_i(s) = \emptyset$ whenever $r \neq s$, and $A(i) = \cup_{k=1}^{p_i} N_i(k)$. Then, KPCC can be formulated as the 0-1 programming problem

$$\begin{aligned} \text{StarIP: Maximize} \quad & \sum_{j \in V} c_j x_j \\ \text{Subject to:} \quad & \sum_{j \in V} a_j x_j \leq b \\ & \sum_{j \in N_i(k)} x_j \leq |N_i(k)|(1 - x_i) \text{ for } k = 1, 2, \ldots p_i, i = 1, 2, \ldots n \\ & x_j \in \{0, 1\} \text{ for all } j \in V \end{aligned} \tag{3}$$

The above formulation reduces to the standard integer programming formulation of the KPCC when each $N_i(k)$ is singleton, i.e., $p_i = |A(i)|$ for all i. When $p_i = 1$ for all i then $N_i(1) = A(i)$, and the above integer program reduces to

$$\begin{aligned} \text{MaxStarIP: Maximize} \quad & \sum_{j \in V} c_j x_j \\ \text{Subject to:} \quad & \sum_{j \in V} a_j x_j \leq b \\ & \sum_{j \in A(i)} x_j \leq |A(i)|(1 - x_i) \text{ for } i = 1, 2, \ldots n \\ & x_j \in \{0, 1\} \text{ for all } j \in V \end{aligned} \tag{4}$$

Thus, our general integer programming framework StarIP provides a class of integer programming formulations of KPCC that subsumes two well-studied formulations.

Theorem 6. *Among all of the choices of the partition $N_i(1), N_i(2), \ldots, N_i(p_i)$ for StarIP, the standard formulation is the strongest and MaxStarIP is the weakest in terms of their LP relaxation values.*

Proof. Let $\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)$ be any feasible solution to the LP relaxation of the standard formulation. Then,

$$x_i^* + x_j^* \leq 1 \text{ for all } (i, j) \in E \tag{5}$$

We successfully show that $\mathbf{x}^*$ is a feasible solution to the LP relaxation of StarIP for any partition $N_i(1), N_i(2), \ldots, N_i(p_i)$. Choose a set $N_i(k)$ for the vertex i. Now, by adding the edge inequalities from (5) for all $(i, j) \in E$ such that $j \in N_i(k)$, we obtain

$$\sum_{j \in N_i(k)} (x_i^* + x_j^* \leq 1) = \sum_{j \in N_i(k)} x_j^* \leq |N_i(k)|(1 - x_i^*).$$

Thus, any feasible solution to the standard integer programming formulation of the KPCC is also a solution to the StarIP. Thus, the LP relaxation of the standard formulation is the strongest among the LP relaxation of any formulation within the class of StarIP. The second part of the theorem can be proved in an analogous way. □

Note that, for the LP relaxation of StarIP (for any choice of the partition of $A(i)$), $x_i = \frac{1}{2}$ for all i as a feasible solution, provided that $\frac{1}{2} \sum_{j \in V} a_j \leq b$. One way to strengthen or tighten StarIP is by computing upper bounds on the stability number of the subgraph $G(i, k)$ induced by $N_i(k)$. For example, if this subgraph is perfect, we can calculate the stability number exactly. If α is the largest cardinality of a matching in this graph, then $|N_i(k)|$ can be replaced by $|N_i(k)| - \alpha$. Another way to tighten the formulation StarIP is as follows: let $\gamma(i, k)$ be the objective function value of the LP relaxation of the standard formulation of KPCC restricted to $G(i, k)$, where $c_i = 1$ for all i. Then, $|N_i(k)|$ can be replaced by $\lfloor \gamma(i, k) \rfloor$. This number can be viewed as a fractional budgeted stability number. We can use other similar upper bounds to strengthen StarIP. For a review of different upper

bounds that can be easily calculated, we refer to [43]. By tightening StarIP this way, we can obtain a better formulation for KPCC.

Let $A'(i) = \{j \in V : (i,j) \in E, j < i\}$ and $N'_i(1), N'_i(2), \ldots, N'_i(p_i)$ be a given partition of $A'(i)$. We can replace $N_i(k)$ by $N'_i(k)$ in StarIP to obtain a valid formulation, say StarIP′, of KPCC. Likewise, we can replace $A(i)$ by $A'(i)$ in MaxStarIP to obtain a valid formulation, say MaxStarIP′, for KPCC. The formulation MaxStarIP′ has been studied by Hifi et al. [6]. The coefficient matrices of these modified formulations are sparser. However, tightening the resulting formulations by using upper bounds as discussed above are likely to yield weaker LP relaxations, in general, compared to tightening the corresponding formulation of StarIP and MaxStarIP. Moreover, any possible computational advantage achieved by sparsity over the suspected weaker LP relaxation needs to be analyzed experimentally.

Based on the ideas used in Theorem 2, we now present a single integer programming formulation of KPCC on $K_{m,n}$.

$$\begin{aligned}
\text{KPCC}_{m,n}\text{: Maximize} \quad & \sum_{j \in V_1 \cup V_2} c_j x_j \\
\text{Subject to:} \quad & \sum_{j \in V_1} a_j x_j \leq b \\
& \sum_{j \in V_2} a_j x_j \leq b \\
& \sum_{j \in V_1} x_j \leq m(1 - x_i) \text{ for } i \in V_2 \qquad (6) \\
& \sum_{j \in V_2} x_j \leq n(1 - x_i) \text{ for } i \in V_1 \qquad (7) \\
& x_j \in \{0,1\} \text{ for all } j \in V_1 \cup V_2
\end{aligned}$$

Note that constraint (6) guarantees that if any $x_i = 1$ for $i \in V_2$ then $x_i = 0$ for all $i \in V_1$. Likewise, constraint (7) guarantees that if any $x_i = 1$ for $i \in V_1$ then $x_i = 0$ for all $i \in V_2$.

4.1. Experimental Analysis

The formulation MaxStarIP′ has been studied and used in algorithms to solve KPCC [6]. There are also effective branch-and-bound algorithms to solve KPCC for general graphs [6,28,44]. However, to the best of our knowledge, there is no computational study available that compares the different formulations of KPCC. Note that the complexity of KPCC can vary significantly depending on the structure of the underlying graph and the level of tightness of the budget constraint. Also, tight efficiently computable upper bound estimates on the stability number of a graph can affect the efficacy of our formulations StarIP and MaxStarIP. The following Table 1 provides some easily computable upper bounds on the stability number of the graph $G = (V, E)$.

In the table above, Δ denotes the maximum degree and δ denotes the minimum degree of vertices in the graph $G = (V, E)$.

Recall that the integer programming formulation MaxStarIP belongs to the class StarIP. Therefore, it can be strengthened by replacing $|A(i)|$ with an upper bound on the stability number of the subgraph $G(i)$ of G induced by $A(i)$. When MaxStarIP is strengthened using the bound UBk obtained from $G(i)$ to replace $|A(i)|$, $k = 1, 2, 3, 4, 5, 6$, we denote the resulting MaxStarIP formulation by MaxStarIPk. Thus, in addition to MaxStarIP, we have the strengthened formulations MaxStarIP1, MaxStarIP2, MaxStarIP3, MaxStarIP4, MaxStarIP5, and MaxStarIP6. The major goals of the experimental analysis were to:

1. Compare the relative performance of the standard KPCC formulation given in the introduction, MaxStarIP, and its variations MaxStarIPk, $k = 1, 2, 3, 4, 5, 6$ using the general purpose solver GUROBI.
2. Study the impact of the tightness of the budget constraints on the formulations that we compare.
3. Study the impact of sparsity of G on the formulations that we compare.

4. Explore the heuristic value of the formulations.

It may be noted that UB1 will be at least as tight as UB2 and that UB4 applies to a connected graph. For these reasons, we will not use UB2 and UB4 in our experiments.

Table 1. Easily computable stability number bounds.

	Bound	Name
UB1	$\lfloor \gamma(G) \rfloor$	fractional budgeted stability number
UB2	$\lfloor \alpha_f(G) \rfloor$	fractional stability number [43]
UB3	$\lfloor \frac{1}{2} + \sqrt{\frac{1}{4} + \lvert V \rvert^2 - \lvert V \rvert - 2\lvert E \rvert} \rfloor$	Hansen [45]
UB4	$\lfloor \frac{(\Delta-1)n+1}{\Delta} \rfloor$	Borg [46]
UB5	$\lfloor \lvert V \rvert - \frac{\lvert E \rvert}{\Delta} \rfloor$	Kwok [47]
UB6	$\lvert V \rvert - \delta$	Minimum degree [43]

All of the computational experiments were carried out on a PC with the Windows 10 Enterprise 64-bit operating system, Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz, and 16.0 GB of memory. The calling program was written in Python (Python 3.6.1) and called GUROBI 9.0 using the interface gurobipy. The time limit for GUROBI was set to 3600 s. Most of the test problems were solved to optimality within this time limit, except some large or structured instances.

4.2. Benchmark Instances

Our experiments used benchmark instances selected from Bettinelli et al. [28] and instances generated using the DIMACS clique graph library [48]. The instances of Bettinelli et al. [28] have three data sets: R1, R2, and R10. Each data set consists of eight classes, and each class has many instances. For the data set R1, in the first four classes, the items have a uniformly distributed weight from the range [20, 100], and the knapsack capacity is set to $b = 150$. The number n of items is 120, 250, 500, and 1000, respectively. The last four classes have uniformly distributed weights from [250, 500], and the knapsack capacity is set to $b = 1000$. The conflict graphs were generated randomly. For the data set R1, classes 1 to 4 have a total of 360 instances and classes 5 to 8 have a total of 360 instances. Thus, the data set R1 has a total of 720 instances. The data set R3 is similar to R1 except that the budget b is set to $3 \times 150 = 450$. Likewise, the data set R10 is almost identical to R1 except that the budget b is set to $10 \times 150 = 1500$. For details on these instances, we refer to the paper [28].

For our computational experiments, we selected instances from the first four classes of each of the data sets R1, R3, and R10. We categorized these instances further into small, medium, and large. The small instances consist of conflict graphs with 120 to 250 vertices, the medium instances consist of conflict graphs with 500 vertices, and the large instances consist of conflict graphs with 1000 vertices. From each category, we selected 120 instances using a prescribed selection rule. We refer to the aforementioned class of instances as random data instances.

The DIMACS clique graph library consists of unweighted graphs. We generated two sets of KPCC benchmarks out of these instances, called positively correlated problems and negatively correlated problems. For positively correlated problems, the costs and weights have the same ordering, whereas for the negatively correlated problems, the costs and weights have opposite ordering. We used graphs that consist of 171 to 1024 vertices. These instances are further classified into small, medium, and large sizes. The small instances consist of graphs with 171 to 378 vertices, the medium instances consist of graphs with 400 to 500 vertices, and the large instances consist of 700 to 1024 vertices. There are 23 small instances, 24 medium instances, and 13 large instances. The DIMACS graphs were originally generated as clique benchmarks. To obtain stable set instances, we took the complement of these graphs. Let $G = (V, E)$ be such a complement graph. For each $i \in V$, we generated the cost c_i and the weight a_i as uniformly distributed random integers in the range $[1, |V|]$. To obtain positively correlated data, we first sort the vectors **c** and **a** in non-decreasing order

and then apply the same random index permutation to both vectors. Negatively correlated instances are constructed in a similar manner but with opposite ordering. The budgets for the correlated instances are set as follows. First, we generate a random subset S of the vertices of G such that the cardinality of S is the known size of the maximum cardinality stable set on G and set $b = \sum_{i \in S} a_i$. Then, instances are generated by setting the budget equal to the value b scaled by the factors 0.25, 0.50, and 0.75. The instances we used in our experiments are available at https://github.com/jasdeepdhahan/kpcc_instances.git.

4.3. Experimental Results

Let us now discuss the experimental outcomes of our analysis. First, we compared the standard KPCC formulation given in the introduction to MaxStarIP and its variations MaxStarIP1, MaxStarIP3, MaxStarIP5, and MaxStarIP6 using all of the test instances. Table 2 provides some insight into the relative strength of our formulations based on average running times (in seconds). The numbers marked in bold letters correspond to the lowest computational time taken. The experimental results disclose that there is no consistently superior formulation of KPCC among the ones we tested.

Table 2. Average running times (in seconds).

Data Set	Standard	MaxStarIP	MaxStarIP1	MaxStarIP3	MaxStarIP5	MaxStarIP6
Random Small	**3.9525**	4.1754	4.3713	9.2690	8.9970	9.2855
Random Medium	547.0358	546.1085	**532.7913**	542.4379	562.4661	540.5295
Random Large	1165.9380	1232.1670	**1116.0310**	1267.2090	1235.8030	1284.1800
Positive Small	0.4756	0.4833	0.4683	0.4874	0.4883	**0.4606**
Positive Medium	117.7284	108.5378	112.0652	**105.2428**	105.8007	109.7333
Positive Large	499.7824	506.0453	417.1473	418.3175	446.0069	**407.9784**
Negative Smal	**1.3642**	1.9495	1.7018	1.9759	1.7312	1.6509
Negative Medium	427.9115	499.4719	443.6538	431.8717	432.7311	**422.1719**
Negative Large	1866.9380	1937.1970	1780.2640	1723.5700	**1698.0620**	1742.0950

The efficacy of various formulations could also be dependent on how tight the budget constraint is and what special structure the underlying graph has. Table 3 presents the experimental results after classifying the budget constraint in the test instances as tight, very tight, and moderately tight. Table 4 presents the experimental results after classifying the random test instances based on tightness of the budget constraints as well as the density of the underlying graph. We sometimes refer to a KPCC instance as sparse or dense. A *sparse (dense)* KPCC instance has a sparse (dense) underlying graph. The results on the correlated test instances are classified only with respect to budget tightness since the density of the underlying graphs was uniform and generally sparse. These results are presented in Table 5.

After this categorized analysis, the standard formulation appears as the preferred formulation overall, if one is constrained to recommend only one formulation. However, it may be noted that MaxStarIP appears to be more suited for sparse small instances with the tightest and loosest budgets. In addition, MaxStarIP outperforms the standard formulation on sparse medium and large instances with the tightest budgets. For our medium and large instances, as the underlying graph density increased the standard formulation became more robust than MaxStarIP. It is also important to point out that 8 out of 120 medium-sized instances were not solved to optimality within the time limit provided, either by the standard formulation or by the MaxStarIP type formulations. In addition, 30 of the 120 large instances with the loosest budget were not solved to optimality within the time limit specified by either of the aforementioned formulations.

Now, let us compare the standard formulation and MaxStarIP type formulations over the correlated data. We found that the positively correlated instances were easier to solve than the negatively correlated instances. Increasing the budget within limits generally increased the running time of our formulations. Large negatively correlated instances often hit the time limit set for the experiments. The standard formulation generally outperformed

MaxStarIP over our negatively correlated data set. We further observed that MaxStarIP is slightly better than the standard formulation for negatively correlated large instances with the loosest budget. For positively correlated data, MaxStarIP is best suited for medium instances and the standard formulation is the strongest for our small and large instances.

It may also be noted that the standard formulation is the most robust in terms of running times, followed by MaxStarIP1, and there is no clear formulation that wins third place. MaxStarIP1 outperformed our other KPCC formulations over denser and larger random instances, while the standard formulation showed superiority on sparser large random instances. Each of the formulations MaxStartIP, MaxStarIP3, MaxStarIP5, and MaxStarIP6 were found to be the preferred formulation for at least one of the random data set classifications in Table 4. Based on this, we conclude that there is no one formulation that is best suited for our positively correlated data set. The standard formulation is best suited for our medium and small negatively correlated data. In addition, MaxStarIP3 outperformed the other formulations over our larger negatively correlated instances with the tightest budgets, and MaxStarIP5 outperformed the other formulations in Table 5 for the remaining large negatively correlated KPCC instances.

Table 3. Average running times (in seconds) by budget.

Data Set	Budget	Standard	MaxStarIP	MaxStarIP1	MaxStarIP3	MaxStarIP5	MaxStarIP6
Random Small	very tight	0.17025	0.2180	**0.1535**	14.48525	14.46275	14.4683
Random Small	tight	**0.8165**	1.4390	0.98075	0.93225	0.95875	1.0078
Random Small	moderate tight	10.871	**10.86925**	11.97975	12.3895	11.56975	12.3805
Random Medium	very tight	**1.2145**	1.78725	1.602	1.89275	1.769	1.87975
Random Medium	tight	24.9155	47.09275	**21.021**	51.623	63.196	37.62275
Random Medium	moderate tight	1614.97725	1589.4455	1575.751	1573.798	1622.43325	**1582.0860**
Random Large	very tight	**5.5705**	10.0915	12.59575	10.38025	10.95975	11.4418
Random Large	tight	491.50875	593.444	484.346	806.51925	**796.03675**	837.51325
Random Large	moderate tight	3000.7355	3092.9655	**2851.1515**	2984.729	2900.411	3003.5857
Positive Small	very tight	0.2232	**0.1991**	0.2173	0.2155	0.2045	0.2245
Positive Small	tight	0.4123	**0.3786**	0.4532	0.4936	0.4955	0.4745
Positive Small	moderate tight	0.7914	0.8723	0.7345	0.7532	0.7650	**0.6827**
Positive Medium	very tight	5.4561	4.7843	3.5357	3.6765	3.6857	3.7887
Positive Medium	tight	47.9278	33.6848	35.3148	32.3804	**29.4691**	32.3991
Postive Medium	moderate tight	299.8013	287.1443	297.34522	**279.6713**	284.2474	293.0122
Positive Large	very tight	**29.4941**	35.5012	41.20824	45.17471	47.0088	41.72882
Positive Large	moderate tight	1093.12647	1082.2853	918.3300	919.5976	959.8629	**908.9182**
Positive Large	tight	376.7265	400.34941	291.9035	290.1800	331.1488	**273.2882**
Negative Small	very tight	**0.3973**	0.4814	0.52410	0.5064	0.4745	0.4614
Negative Small	tight	**0.9173**	1.2236	1.17409	1.2859	1.13911	1.3295
Negative Small	moderate tight	**2.7782**	4.1436	3.4073	4.1354	3.5800	3.1618
Negative Medium	very tight	**6.5067**	10.5179	7.8883	9.2575	9.2071	8.8704
Negative Medium	tight	**266.1083**	292.6083	276.6429	274.4096	299.2258	270.0600
Negative Medium	moderate tight	1011.11958	1195.2896	1046.4300	1011.9479	989.7604	**987.5854**
Negative Large	very tight	721.0029	857.0400	667.3035	**649.35523**	666.8012	658.71412
Negative Large	tight	2095.2776	2210.1188	2010.5912	1965.06882	**1879.2935**	1955.7835
Negative Large	moderate tight	2784.5333	2744.4318	2662.8976	2556.2859	**2548.0912**	2611.7882

Table 4. Average running times (in seconds) by budget and density.

Data Set	Budget	Density	Standard	MaxStarIP	MaxStarIP1	MaxStarIP3	MaxStarIP5	MaxStarIP6
Random Small	very tight	0.10–0.30	0.0657	0.04500	**0.0421**	40.7564	40.75	40.7564
Random Small	tight	0.10–0.30	**0.1957**	0.2893	0.2364	0.2621	0.2986	0.2707
Random Small	moderate tight	0.10–0.30	12.5571	**12.3421**	20.2229	17.135	17.52	18.2271
Random Small	very tight	0.40–0.60	0.1557	0.1700	**0.1486**	0.1614	0.155	0.1586
Random Small	tight	0.40–0.60	**0.7914**	1.6179	0.8986	1.0443	0.9907	1.0564
Random Small	moderate tight	0.40–0.60	15.6157	15.7507	**11.5864**	12.2021	11.8821	12.4986
Random Small	very tight	0.70–0.90	0.3092	0.4758	**0.2892**	0.5467	0.4867	0.4933
Random Small	tight	0.70–0.90	**1.5700**	2.5717	1.945	1.5833	1.6917	1.8108
Random Small	moderate tight	0.70–0.90	3.0409	3.1609	**2.5527**	5.5382	2.9364	5.3027
Random Medium	very tight	0.10–0.30	0.3307	0.2493	0.2779	0.2471	**0.2386**	0.2421
Random Medium	tight	0.10–0.30	**1.2657**	1.8571	1.6014	2.3179	1.7207	1.7429
Random Medium	moderate tight	0.10–0.30	1807.6836	1792.2193	1933.2079	1802.3836	1830.6364	**1778.0714**
Random Medium	very tight	0.40–0.60	0.91	0.8462	0.8869	0.8215	0.8654	**0.8023**
Random Medium	tight	0.40–0.60	**16.8238**	29.1508	18.8531	21.9731	23.0215	21.1769
Random Medium	moderate tight	0.40–0.60	2456.7877	2391.1946	**2309.3023**	2380.9877	2421.0185	2427.7085
Random Medium	very tight	0.70–0.90	**2.4708**	4.3846	3.7431	4.7362	4.3208	4.7208
Random Medium	tight	0.70–0.90	58.4762	113.75	**44.1023**	134.3708	169.5746	92.7085
Random Medium	moderate tight	0.70–0.90	565.6369	569.3246	**457.2462**	520.4392	599.6292	525.4023
Random Large	very tight	0.10–0.30	1.3021	1.2186	1.2957	1.1879	1.1857	**1.1764**
Random Large	tight	0.10–0.30	**7.3279**	12.9400	8.505	13.2321	8.8986	9.1857
Random Large	moderate tight	0.10–0.30	**2475.0386**	2547.7179	2487.0064	2504.4529	2519.53	2556.8364
Random Large	very tight	0.40–0.60	**4.3062**	4.6062	5.6438	4.4131	4.5692	4.4046
Random Large	tight	0.40–0.60	**211.7485**	328.7346	303.8569	476.9715	466.3538	451.49
Random Large	moderate tight	0.40–0.60	3600.2546	3601.3262	3600.0577	3600.0585	3600.0546	3600.0592
Random Large	very tight	0.70–0.90	**11.4315**	25.1323	31.7169	26.2469	27.8762	29.5338
Random Large	tight	0.70–0.90	1292.6946	1483.3115	**1177.2792**	1990.3762	1973.4069	2115.5815
Random Large	moderate tight	0.70–0.90	2967.3515	3171.7946	**2494.4015**	2886.62	2610.9469	2888.2269

Table 5. Average running times (in seconds) by budget and density.

Data Set	Budget	Density	Standard	MaxStarIP	MaxStarIP1	MaxStarIP3	MaxStarIP5	MaxStarIP6
Positive Small	very tight	Sparse	0.2232	**0.1991**	0.2173	0.2155	0.2045	0.2245
Positive Small	tight	Sparse	0.4123	**0.3786**	0.4532	0.4936	0.4955	0.4745
Positive Small	moderate tight	Sparse	0.7914	0.8723	0.7345	0.7532	0.7650	**0.6827**
Positive Medium	very tight	Sparse	5.4561	4.7843	3.5357	3.6765	3.6857	3.7887
Positive Medium	tight	Sparse	47.9278	33.6848	35.3148	32.3804	**29.4691**	32.3991
Positive Medium	moderate tight	Sparse	299.8013	287.1443	297.34522	**279.6713**	284.2474	293.0122
Positive Large	very tight	Sparse	**29.4941**	35.5012	41.20824	45.17471	47.0088	41.72882
Positive Large	moderate tight	Sparse	1093.12647	1082.2853	918.3300	919.5976	959.8629	**908.9182**
Positive Large	tight	Sparse	376.7265	400.34941	291.9035	290.1800	331.1488	**273.2882**
Negative Small	very tight	Sparse	**0.3973**	0.4814	0.52410	0.5064	0.4745	0.4614
Negative Small	tight	Sparse	**0.9173**	1.2236	1.17409	1.2859	1.13911	1.3295
Negative Small	moderate tight	Sparse	**2.7782**	4.1436	3.4073	4.1354	3.5800	3.1618
Negative Medium	very tight	Sparse	**6.5067**	10.5179	7.8883	9.2575	9.2071	8.8704
Negative Medium	tight	Sparse	**266.1083**	292.6083	276.6429	274.4096	299.2258	270.0600
Negative Medium	moderate tight	Sparse	1011.11958	1195.2896	1046.4300	1011.9479	989.7604	**987.5854**
Negative Large	very tight	Sparse	721.0029	857.0400	667.3035	**649.35523**	666.8012	658.71412
Negative Large	tight	Sparse	2095.2776	2210.1188	2010.5912	1965.06882	**1879.2935**	1955.7835
Negative Large	moderate tight	Sparse	2784.5333	2744.4318	2662.8976	2556.2859	**2548.0912**	2611.7882

The frequency distribution of the fastest running times can provide further insight into the complexity of KPCC that averaging out the running times could not provide. For this analysis, we consider only the test instances that were solved to optimality within the time limit specified. In the event of a tie, any formulations in a tie for the fastest running time were considered to be equally good. From Table 6, we observe that the standard formulation is the preferred formulation followed by MaxStarIP1, and there is no clear winner for third place, based on frequency analysis.

Table 6. Frequency distribution of fastest running times.

Data Set	Standard	MaxStarIP	MaxStarIP1	MaxStarIP3	MaxStarIP5	MaxStarIP6
Random Small	**62**	12	38	17	25	14
Random Medium	**41**	14	29	9	17	17
Random Large	**41**	6	21	6	15	11
Positive Small	**22**	2	9	2	3	4
Positive Medium	**16**	15	9	8	12	12
Positive Large	**20**	5	5	6	2	11
Negative Small	**35**	5	10	3	8	9
Negative Medium	**23**	4	16	6	9	10
Negative Large	**15**	1	3	4	8	3

5. Conclusions

In this paper, we studied the KPCC from various points of views. After a thorough literature survey on the topic, we focussed our attention on bipartite conflict graphs. For a complete bipartite (multipartite) graphs, the problem is shown to be NP-hard but solvable in pseudo-polynomial time and admits FPTASs. Extensions of these results to more general classes of graphs are also presented. We then introduced an integer programming model for KPCC with general conflict graphs, which generalizes and unifies existing formulations. The strength of the LP relaxations of these formulations are analyzed and also discussed in different ways to tighten them. An experimental comparison of our models is also presented to assess their relative strengths. This analysis disclosed various strong and weak points of our formulations and their linkages to the structure of the problem data. The analysis shows some preference towards the standard formulation; none of the formulations uniformly dominated others in all categories of the test data. In any case, the insights generated could be used effectively in the design of special purpose algorithms for KPCC involving some learning components. Such approaches proved useful in the case of other combinatorial optimization problems (e.g., [49]). This study can be followed on with future work.

Author Contributions: Conceptualization, A.P.P.; methodology, A.P.P. and J.D.; software, J.D.; validation, J.D.; formal analysis, A.P.P. and J.D.; investigation, A.P.P. and J.D.; resources, A.P.P.; data curation, J.D.; writing – original draft preparation, J.D.; writing—review and editing, A.P.P.; visualization, J.D.; supervision, A.P.P.; project administration, A.P.P.; funding acquisition, A.P.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by an NSERC discovery grant awarded to Abraham P. Punnen.

Data Availability Statement: Our data is available to the public from " https://github.com/jasdeepdhahan/kpcc_instances.git".

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

References

1. Bandyapadhyay, S. A variant of the maximum weight independent set problem. *arXiv* **2014**, arXiv:1409.0173.
2. Kalra, T.; Mathew, R.; Pal, S.P.; Pandey, V. Maximum weighted independent with a budget. In Proceedings of the Algorithms and Discrete Applied Mathematics: Third International Conference, CALDAM 2017, Sancoale, Goa, India, 16–18 February 2017; Gaur, D., Narayanaswamy, N., Eds.; Proceedings 3; Springer: Cham, Switzerland, 2017; Volume 10156, pp. 254–266.
3. Milanic, M.; Monnot, J. The exact weighted independent set problem in perfect graphs and related classes. *Electron. Notes Discret. Math.* **2009**, *35*, 317–322. [CrossRef]
4. Akeb, H.; Hifi, M.; Mounir, M. Local branching-based algorithms for the disjunctively constrained knapsack problem. *Comput. Ind. Eng.* **2011**, *60*, 811–820. [CrossRef]
5. Hifi, M.; Michrafy, M. A reactive local search algorithm for the disjunctively constrained knapsack problem. *J. Oper. Res. Soc.* **2006**, *57*, 718–726. [CrossRef]
6. Hifi, M.; Michrafy, M. Reduction strategies and exact algorithms for the disjunctively constrained knapsack problem. *Comput. Oper. Res.* **2007**, *34*, 2657–2673. [CrossRef]
7. Hifi, M.; Otmani, N. An algorithm for the disjunctively constrained knapsack problem. *Int. J. Oper. Res.* **2012**, 13, 22–43. [CrossRef]
8. Hifi, M.; Saleh, S.; Wu, L.; Chen, J. A hybrid guided neighborhood search for the disjunctively constrained knapsack problem. *Cogent Eng.* **2015**, *2*, 1068969. [CrossRef]

9. Hifi, M. An iterative rounding search-based algorithm for the disjunctively constrained knapsack problem. *Eng. Optim.* **2014**, *46*, 1109–1122. [CrossRef]
10. 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.
11. Carrabs, F.; Cerrone, C.; Pentangelo, R. A multiethnic genetic approach for the minimum conflict weighted spanning tree problem. *Networks* **2019**, *74*, 134–147. [CrossRef]
12. Carrabs, F.; Cerulli, R.; Pentangelo, R.; Raiconi, A. Minimum spanning tree with conflicting edge pairs: A branch-and-cut approach. *Ann. Oper. Res.* **2021**, *298*, 65–78. [CrossRef]
13. Darmann, A.; Pferschy, U.; Schauer, J. Determining a minimum spanning tree with disjunctive constraints. In *Lecture Notes in Computer Science*; including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics; Springer: Berlin/Heidelberg, Germany, 2009; pp. 414–423.
14. Darmann, A.; Pferschy, U.; Schauer, J.; Woeginger, G.J. Paths, trees and matchings under disjunctive constraints. *Discret. Appl. Math.* **2011**, *159*, 1726–1735. [CrossRef]
15. Elhedhli, S.; Li, L.; Gzara, M.; Naoum-Sawaya, J. A branch-and-price algorithm for the bin packing problem with conflicts. *INFORMS J. Comput.* **2011**, *23*, 404–415. [CrossRef]
16. Gendreau, M.; Laport, G.; Semet, F. Heuristics and lower bounds for the bin packing problem with conflicts. *Comput. Oper. Res.* **2004**, *31*, 347–358. [CrossRef]
17. Oncan, T.; Zhang, R.; Punnen, A.P. The minimum cost perfect matching problem with conflict pair constraints. *Comput. Oper. Res.* **2013**, *40*, 920–930 [CrossRef]
18. Pferschy, U.; Schauer, J. The maximum flow problem with disjunctive constraints. *J. Comb. Optim.* **2013**, *26*, 109–119. [CrossRef]
19. Zhang, R.; Kabadi, S.N.; Punnen, A.P. The minimum spanning tree problem with conflict constraints and its variations. *Discret. Optim.* **2011**, *8*, 191–206. [CrossRef]
20. Samer, P.; Urrutia, S. A branch and cut algorithm for minimum spanning trees under conflict constraints. *Optim. Lett.* **2014**, *9*, 41–55. [CrossRef]
21. Pisinger, D.; Sigurd, M. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS J. Comput.* **2007**, *19*, 36–51. [CrossRef]
22. Sadykov, R.; Vanderbeck, F. Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS J. Comput.* **2013**, *25*, 244–255. [CrossRef]
23. Fernandes-Muritiba, A.E.; Iori, M.; Malaguti, E.; Toth, P. Algorithms for the bin packing problem with conflicts. *INFORMS J. Comput.* **2010**, *22*, 401–415. [CrossRef]
24. Chandra, A.K.; Hirschberg, D.S.; Wong, C.K. Approximate algorithms for the knapsack problem and its generalizations. In *IBM Research Report*; RC56l6; IBM T. J. Watson Research Center: New York, NY, USA, 1975.
25. Nauss, R.M. *The 0-1 Knapsack Problem with Multiple Choice Constraints*; University of Missouri-St. Louis: St. Louis, MO, USA, 1975; (Revised in 1976).
26. Ibaraki, T.; Hasegawa, T.; Teranaka, K.; Iwase, J. The multiple choice knapsack problem. *J. Oper. Res. Soc. Jpn.* **1978**, *21*, 59–93.
27. Bar-Noy, A.; Bar-Yehuda, R.; Freund, A.; Naor, J.; Schieber, B. A unified approach to approximating resource allocation and scheduling. *J. ACM* **2001**, *48*, 1069–1090. [CrossRef]
28. Bettinelli, A.; Cacchiani, V.; Malaguti, E. A branch-and-bound algorithm for the knapsack problem with conflict graph. *INFORMS J. Comput.* **2017**, *29*, 457–473. [CrossRef]
29. Salem, M.B.; Taktak, R.; Mahjoub, A.R.; Ben-Abdallah, H. Optimization algorithms for the disjunctively constrained knapsack problem. *Soft Comput.* **2018**, *22*, 2025–2043. [CrossRef]
30. Salem, M.B.; Hanafi, S.; Taktak, R.; Ben-Abdallah, H. Probabilistic Tabu search with multiple neighborhoods for the Disjunctively Constrained Knapsack Problem. *RAIRO-Oper. Res.* **2017**, *51*, 627–637. [CrossRef]
31. Pferschy, U.; Schauer, J. The knapsack problem with conflict graphs. *J. Graph Algorithms Appl.* **2009**, *13*, 233–249. [CrossRef]
32. Pferschy, U.; Schauer, J. Approximation of knapsack problems with conflict and forcing graphs. *J. Comb. Optim.* **2017**, *33*, 1300–1323. [CrossRef]
33. Milanic, M.; Monnot, J. The complexity of the exact weighted independent set problem. In *Combinatorial Optimization-Theoretical Computer Science: Interfaces and Perspectives*; Wiley-ISTE: NewYork, NY, USA, 2008; pp. 393–432.
34. Gabrel, V. Dantzig-Wolfe Decomposition for Linearly Constrained Stable Set Problem; hal-00116732; France 2006. Available online: https://hal.science/hal-00116732/ (accessed on 23 February 2024)
35. Atamtürk, A.; Nemhauser, G.L.; Savelsbergh, M.W.P. Conflict graphs in solving integer programming problems. *Eur. J. Oper. Res.* **2000**, *121*, 40–55. [CrossRef]
36. Gallo, G.; Hammer, P.; Simeone, B. Quadratic knapsack problems. In Combinatorial Optimization; Mathematical Programming Studies; Padberg, M., Ed.; Springer: Berlin, Germany, 1980; Volume 12, pp. 132–149.
37. Hammer, P.L.; Hansen, P.; Simone, B. Roof duality, complementations, and persistency in quadratic 0–1 optimization. *Math. Program.* **1984**, *28*, 121–155. [CrossRef]
38. Punnen, A.P. (Ed.) Introduction to QUBO. In *The Quadratic Unconstrained Binary Optimization Problem: Theory, Algorithms, and Applications*; Springer: Cham, Switzerland, 2022.

39. Punnen, A.P. (Ed.) *The Quadratic Unconstrained Binary Optimization Problem: Theory, Algorithms, and Applications*; Springer: Cham, Switzerland, 2022.
40. Deineko, V.G.; Woeginger, G.J. A well-solvable special case of the bounded knapsack problem. *Oper. Res. Lett.* **2011**, *39*, 118–120. [CrossRef]
41. Bonamy, M.; Dabrowski, K.K.; Feghali, C.; Johnson, M.; Paulusma, D. Recognizing Graphs Close to Bipartite Graphs. In *42nd International Symposium on Mathematical Foundations of Computer Science (MFCS 2017)*; Leibniz International Proceedings in Informatics; Larsen, K.G., Bodlaender, H.L., Raskin, J.-F., Eds.; Schloss Dagstuhl-Leibniz-Zentrum für Informatik GmbH: Wadern, Germany, 2017; pp. 70:1–70:14.
42. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W.H. Freeman Co.: New York, NY, USA, 1979.
43. Willis, W. Bounds for the Independence Number of a Graph. Master's Thesis, College of Humanities and Sciences, Virginia Commonwealth University, Richmond, VA, USA, 2011.
44. Li, J.; Lan, Y.; Chen, F.; Han, X.; Blazewicz, J. A Fast Algorithm for Knapsack Problem with Conflict Graph. *Asia-Pac. J. Oper. Res.* **2021**, *38*, 2150010. [CrossRef]
45. Hansen, P. Degrés et nombre de stabilité d'un graphe. *Cah. Centre Études Rech. Opér.* **1975**, *17*, 213–220.
46. Borg, P. A sharp upper bound for the independence number. *arXiv* **2010**, arXiv:1007.5426.
47. West, D. *Introduction to Graph Theory*, 2nd ed.; Prentice Hall Inc.: Upper Saddle River, NJ, USA, 2001.
48. Rossi, R.; Ahmed, N. The Network Data Repository with Interactive Graph Analytics and Visualization. *Proc. AAAI* **2015**, *29*, 4292–4293. [CrossRef]
49. Karapetyan, D.; Punnen, A.P.; Parkes, A.J. Markov chain methods for the bipartite Boolean quadratic programming problem. *Eur. J. Oper. Res.* **2017**, *260*, 494–506. [CrossRef]