Stronger ILP Models for Maximum Induced Path

Fritz Bökler
Theoretical Computer Science, Osnabrück University, Germany
fboekler@uni-osnabrueck.de

Markus Chimani
Theoretical Computer Science, Osnabrück University, Germany
markus.chimani@uni-osnabrueck.de

Mirko H. Wagner
Theoretical Computer Science, Osnabrück University, Germany
mirwagner@uni-osnabrueck.de

Tilo Wiedera
Theoretical Computer Science, Osnabrück University, Germany
tilo.wiedera@uni-osnabrueck.de

Abstract

Given a graph $G = (V, E)$, the Longest Induced Path problem asks for a maximum cardinality node subset $W \subseteq V$ such that the graph induced by W is a path. It is a long established problem with applications, e.g., in network analysis. We propose two novel integer linear programming (ILP) formulations for the problem and discuss algorithms to implement them efficiently. Comparing them with known formulations from literature, we show that they are both beneficial in theory, yielding stronger relaxations. Furthermore, we show that our best models yield running times faster than the state-of-the-art by orders of magnitudes in practice.

2012 ACM Subject Classification Mathematics of computing → Paths and connectivity problems; Theory of computation → Integer programming

Keywords and phrases longest induced path, ILP, polyhedral analysis, experimental algorithmics

Digital Object Identifier 10.4230/LIPIcs.submitted.2019.0

1 Introduction

Let $G = (V, E)$ be an undirected graph, and $W \subseteq V$ a node subset. The *induced graph* $G[W]$ contains exactly those edges of G whose incident nodes are both in W. If $G[W]$ is a path—i.e., we may order W such that there is an edge between any two subsequent nodes, but no further edges—it is called an *induced path*. The problem of finding an induced path of maximum length is called Longest Induced Path and known to be NP-complete [5]. In fact, it is already hard for bipartite graphs, as witnessed by subdividing each edge in a Longest Path instance, another well-known NP-complete problem.

The Longest Induced Path problem has several applications in network analysis, for both social and telecommunication networks. One of the reasons is its relation to the graph *diameter*—the longest among all shortest paths between any two nodes—in a node failure scenario. Removing a node from G may both increase or decrease the graph's diameter. The longest induced path witnesses the largest diameter that may occur by the deletion of any node subset [13]. Observe that the corresponding problem for failing edges is the aforementioned Longest Path problem. The problem of enumerating all induced paths (not only the longest ones) can be used to predict nuclear magnetic resonance [14].

Besides being NP-complete, Longest Induced Path is also W[2]-complete [3] and does not allow a polynomial $\mathcal{O}(|V|^{1/2-\epsilon})$-approximation, $\epsilon > 0$, unless NP = ZPP [2, 9]. On the positive side, it can be solved in polynomial time for several graph classes, e.g., those of bounded mim-width (which includes interval, bi-interval, circular arc, and

permutation graphs) [10] and k-bounded-hole, interval-filament, and other decomposable graphs [6]. Furthermore, some other NP-complete problems, as k-Coloring, $k \geq 5$, [8] and Independent Set [12], are solvable in polynomial time, if the longest induced path has bounded length.

Recently this year, the first non-trivial algorithms to exactly solve the Longest Induced Path problem have been devised in [13]. There, three different ILP (integer linear programming) formulations have been proposed: the first searches for a subgraph with largest diameter; the second utilizes properties derived from the average distance between two nodes of a subgraph; the third models the path as a walk in which no shortcuts can be taken. The authors show that the latter (see later for details) is the most effective in practice.

Contribution. We propose alternative ILP formulations: one based on multi-commodity flow, and one based on cuts and subtour elimination (Section 3). We prove that both our approaches yield strictly stronger ILP formulations than those proposed in [13] (Section 4) and describe a way to strengthen them even further. After discussing some algorithmic considerations (Section 5), we show that our most effective model also in practice out-performs the previously known approaches, often by orders of magnitude (Section 6).

2 Preliminaries

Notation. For $k \in \mathbb{N}$, let $[k] := \{0, \dots, k-1\}$. Throughout this paper, we consider a connected, undirected, simple graph $G = (V, E)$ as our input, and define $n := |V|$. Edges are cardinality-two subsets of V. If there is no ambiguity, we may write uv for edge $\{u, v\}$. Given a graph H, we refer to its nodes (edges) by $V(H)$ ($E(H)$, respectively). For $W \subseteq V$, let $G[W] := (W, \{e \in E : |e \cap W| = 2\})$ denote the node-induced subgraph of G. Given a cycle C in G, a *chord* is an edge connecting two nodes of $V(C)$ that are not neighbors along C.

Linear programming. A *linear program* (LP) consists of a cost vector $c \in \mathbb{Q}^d$ together with a set of linear inequalities, called *constraints*, that define a polyhedron $\mathcal{P}$ in $\mathbb{R}^d$. In polynomial time, we can find a point $x \in \mathcal{P}$ that maximizes the *objective function* $c^\intercal x$. Unless P = NP, this is no longer true when restricting x to have integral components; the so-modified problem is an *integer linear program* (ILP). Conversely, the *LP relaxation* of an ILP is obtained by dropping the integrality constraints on the components of x. The optimal value of an LP relaxation is a dual bound on the ILP's objective; e.g., an upper bound for maximization problems. Typically, there are several ways to *model* a given problem as an ILP. To achieve good practical performance, one aims for models that yield small dimensions and strong dual bounds. This is crucial, as ILP solvers are based on a branch-and-bound scheme that relies on iteratively computing LP relaxations to obtain dual bounds on the ILP's objective. When a model contains too many constraints, it is often sufficient to use only a reasonably sized constraint subset to achieve provably optimal solutions. This allows us to add constraints during the solving process, which is called *separation*. We say that model A is *at least as strong* as model B, if for all instances, the LP relaxation's value of model A is no worse than that of B. If there also exists an instance for which A's LP relaxation yields a tighter bound than that of B, then A is *stronger* than B.

When referring to models, we use the prefix "ILP" and give a short name as subscript. When referring to their respective LP relaxations we write "LP" instead.

Walk-based model (state-of-the-art). The following ILP model, denoted by ILP_{Walk}, was recently presented in [13] (called `A3c` therein). It constitutes the foundation of the fastest known exact algorithm. It models a "timed" walk through the graph that prevents "short-cut" edges. Let T denote an upper bound on the length of the path, i.e., on its number of edges.

$$\max \sum_{t=1}^{T} \sum_{v \in V} x_v^t \tag{1a}$$

$$\text{s.t.} \sum_{v \in V} x_v^t \leq 1 \qquad \forall t \in [T+1] \tag{1b}$$

$$\sum_{t=0}^{T} x_v^t \leq 1 \qquad \forall v \in V \tag{1c}$$

$$\sum_{v \in V} x_v^{t+1} \leq \sum_{v \in V} x_v^{t-1} \qquad \forall t \in [T] \tag{1d}$$

$$x_v^t \leq 1 - \sum_{w \in V : vw \notin E} x_w^{t+1} \qquad \forall v \in V, t \in [T] \tag{1e}$$

$$x_v^t \leq 1 - \sum_{\tau=t+2}^{T} x_j^\tau \qquad \forall v \in V, t \in [T-1] \tag{1f}$$

$$x_v^t \in \{0,1\} \qquad \forall v \in V, t \in [T+1] \tag{1g}$$

For every node $v \in V$ and every point in time $t \in [T+1]$ there is a variable x_v^t that is 1 iff v is visited at time t (1g). In every step at most one vertex can be visited (1b); a vertex can be visited once at most (1c); the time points have to be used consecutively (1d); nodes visited at consecutive time points need to be adjacent (1e); and nodes at non-consecutive time points cannot be adjacent (1f).

However, ILP_{Walk} yields only weak LP relaxations (cf. Section 4). To overcome this, [13] proposes to iteratively solve ILP_{Walk} for increasing values of T until the objective value is less than T. They use the graph's diameter as a lower bound on T to avoid trivial calls. In addition, they add the following supplemental inequalities that are valid if there is a solution of length exactly T: only the first and the last node may have degree 1, and if this holds only for one of them, it shall be the first to break symmetries.

$$x_v^{t+1} = 0 \qquad \forall v \in V \colon \deg(v) = 1, t \in [T-1] \tag{1h}$$

$$\sum_{v \in V : \deg(v)=1} x_v^0 \geq \sum_{v \in V : \deg(v)=1} x_v^T \tag{1i}$$

3 New Models

We present two new ILP models. Instead of requiring a "timed" walk, they are based only on a single decision variable for each edge (and possibly node) specifying whether the respective graph element is in the solution. Most importantly, our models yield stronger LP relaxations than ILP_{Walk} (see Section 4). Thus, while ILP_{Walk} in practice requires to iteratively perform multiple computations for different upper bounds T, our models can be solved via a single ILP computation and without the need of any such bound.

We start with describing a *partial* model ILP_{Base}, which by itself is not sufficient but consitutes the common core of our new models. Afterwards, we will add to it, to obtain full models for Longest Induced Path: ILP_{Cut} and ILP_{Flow}.

For notational simplicity, we augment $G = (V, E)$ to $G^* := (V^* = V \cup \{s\}, E^* = E \cup \{sv\}_{v \in V})$ by adding a new node s that is connected to all nodes of V. Within G^*, we will now look for a longest induced cycle through s, where we ignore induced chords incident to s. Searching for a cycle instead of a path, allows us to homogeneously require that each *selected* edge, i.e., edge in the solution, has exactly two adjacent edges that are also selected.

Let $\delta^*(e) \subset E^*$ denote the edges adjacent to edge e in G^*. The binary x_e-variables in the model indicate whether an edge e is selected. We denote the partial model below by ILP_{Base}:

$$\max \sum_{e \in E} x_e \tag{2a}$$

$$\text{s.t.} \sum_{v \in V} x_{sv} = 2 \tag{2b}$$

$$2x_e \leq \sum_{f \in \delta^*(e)} x_f \leq 2 \qquad \forall e \in E \tag{2c}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E^* \tag{2d}$$

Constraint (2b) requires to select exactly two edges incident with s. To prevent chords, constraints (2c) enforce that any edge e (even if not selected itself) is adjacent to at most two edges of $\mathcal{C}$; if e is selected, precisely two of its adjacent edges need to be selected.

However, the above model is not sufficient: it allows for the solution to consist of multiple disjoint cycles, only one of which contains s. But these cycles have no chords in G, and no edge in G connects any two cycles. To obtain a longest single cycle C through s—yielding the longest induced path $G[V(C) \setminus \{s\}]$—we thus have to forbid additional cycles in the solutions that are not containing s. In other words, we want to enforce that the graph induced by the x-variables is connected. There are three established ways to achieve this: via *cut* or *(generalized) subtour elimination* constraints on the one hand, and via a compact *multi-commodity flow* model on the other hand.

Cut model (and generalized subtour elimination). We define $\delta^*(W) := \{w\bar{w} \in E^* \mid w \in W, \bar{w} \in V^* \setminus W\}$ as the set of edges in the cut induced by $W \subseteq V^*$. For notational simplicity, we may omit braces when referring to node sets of cardinality one. We obtain ILP_{Cut} by adding the below *cut* constraints to ILP_{Base}.

$$\sum_{e \in \delta^*(v)} x_e \leq \sum_{e \in \delta^*(W)} x_e \qquad \forall W \subseteq V, v \in W \tag{3a}$$

These constraints ensure that if a node v is incident to a selected edge (by (2c) there are then two such selected edges), any cut separating v from s contains at least two selected edges, as well. Thus, there are (at least) two edge-disjoint paths between v and s selected. Together with the cycle properties of ILP_{Base}, we can deduce that v and its selected incident edges lie on a common selected cycle with s.

An alternative view leads to *subtour elimination* constraints $\sum_{e \in E : e \subseteq W} x_e \leq |W| - 1$ for $W \subseteq V$, which prohibit cycles not containing s via counting. It is well known that these constraint can be generalized using binary node variables $y_v := \frac{1}{2} \sum_{e \in \delta^*(v)} x_e$ that indicate whether node $v \in V$ participates in the solution (in our case: in the induced path). *Generalized subtour elimination* constraints thus take the form

$$\sum_{e \in E : e \subseteq W} x_e \leq \sum_{w \in W \setminus \{v\}} y_w \qquad \forall W \subseteq V, v \in W. \tag{3b}$$

One expects LP_{Cut} and "LP_{Base} with constraints (3b)" to be equally strong as this is well-known for standard Steiner, TSP, and other related models. Interestingly, in the case of Longest Induced Path there even is a direct one-to-one correspondence between cut constraints (3a) and generalized subtour elimination constraints (3b): By substituting node-variables with their definitions in (3b), we obtain $2 \sum_{e \in E : e \subseteq W} x_e \leq \sum_{w \in W \setminus \{v\}} \sum_{e \in \delta^*(v)} x_e$. A simple rearrangement yields a corresponding cut constraint (3a).

Multi-commodity flow model. In contrast to the above method, flow formulations allow a compact, i.e., polynomially-sized, model. Each node $v \in V$ is assigned a commodity and sends—if v is part of the induced path—two units of flow of this commodity from v to s using only selected edges. This ensures that each node in the solution lies on a common cycle with s. Consider the (nearly bidirected) arc set $A^* := \{(vw), (wv) \mid \{v, w\} \in E\} \cup \{(vs) \mid v \in V\}$ that consists of a directed arc for each possible direction of each edge in E^*, except for arcs leaving s. Let $\delta^*_{\text{out}}(v)$ ($\delta^*_{\text{in}}(v)$) denote the arcs of A^* with source (target) $v \in V$. We use variables z^v_a to model the flow of commodity v over arc $a \in A^*$; we do not need them to be binary. The below model, together with ILP_{Base}, forms ILP_{Flow}.

$$z^v_{(vw)} \leq x_{\{v,w\}} \qquad \forall v \in V, \{v, w\} \in E^* \tag{4a}$$

$$z^v_{(vw)} = x_{\{v,w\}} \qquad \forall v \in V, \{v, w\} \in E^* \tag{4b}$$

$$\sum\nolimits_{a \in \delta^*_{\text{out}}(w)} z^v_a = \sum\nolimits_{a \in \delta^*_{\text{in}}(w)} z^v_a \qquad \forall w, v \in V, w \neq v \tag{4c}$$

$$0 \leq z^v_a \leq 1 \qquad \forall v \in V, a \in A^* \tag{4d}$$

The capacity constraints (4a) ensure that flow is only sent over selected edges. Equations (4b) send the commodities away from their source v, if v is part of the solution; and equations (4c) model flow preservation (up to, but not including, the sink s).

Clique constraints. We can further strengthen both our above models, by introducing a set of additional inequalities. Consider any clique (i.e., complete subgraph) in G. The induced path may contain at most one of its edges:

$$\sum\nolimits_{e \in E : e \subseteq Q} x_e \leq 1 \qquad \forall Q \subseteq V : G[Q] \text{ is a clique.} \tag{5}$$

4 Polyhedral Properties of the LP Relaxations

We can compare the above models w.r.t. the strength of their LP relaxations, i.e., the quality of their dual bounds. Achieving strong dual bounds is a highly relevant goal also in practice: one can expect a lower running time for the ILP solvers in case of better dual bounds since less nodes of the underlying branch-and-bound tree have to be explored.

Since ILP_{Walk} requires *some* upper bound T on the objective value, we can only reasonably compare this model to ours by assuming that the latter are also given this bound as an explicit constraint. (For the most general scenario, one may assume the trivial upper bound $T = n - 1$; but we may also compare the models in a scenario where T is, e.g., infeasibly low.) By construction, no dual bound of any of the considered models will hence yield a worse (i.e., larger) bound than T. As has already been observed in [13], LP_{Walk} will in fact *always* yield this worst case bound:

▶ **Proposition 1** (Proposition 5 from [13]). LP_{Walk} *has objective value* T *for every* $T < n$.

Proof. We set x^t_v to $1/n$ for all $v \in V$ and $t \in [T]$. It is easy to see that this solution is feasible and achieves the claimed objective value. ◀

Note that Proposition 1 is independent of the graph. Given that the longest induced path of a complete graph has length 1, we also conclude that the integrality gap of ILP_{Walk} is unbounded. Furthermore, this shows that LP_{Base} cannot be weaker than LP_{Walk}. We show that already the partial model LP_{Base} is in fact *stronger* than LP_{Walk}. Let therefore $\theta := T - \text{OPT} \in \mathbb{N}$, where OPT is the instance's (integral) optimum value.

▶ **Proposition 2.** *For every $\theta \geq 1$, we have:* $\mathrm{LP}_{\mathrm{Base}}$ *is stronger than* $\mathrm{LP}_{\mathrm{Walk}}$*, and this is witnessed by infinitely many graphs where the former gives bounds less than* 1 *from* OPT *and the latter worst possible bounds* $\mathrm{OPT} + \theta$*.*

Proof. By Proposition 1, $\mathrm{LP}_{\mathrm{Walk}}$ will always attain value $T = \mathrm{OPT} + \theta$. To show the strength claim, it thus suffices to give instances where $\mathrm{LP}_{\mathrm{Base}}$ yields a strictly tighter bound.

Already a star with at least three leaves proves the claim, as $\mathrm{LP}_{\mathrm{Base}}$ guarantees a solution of optimal value 2. However, it can be argued that such graphs and substructures are easy to preprocess. Thus, we prove the claim with a more suitable instance class.

Choose any $\ell \geq 3$, start with two nodes x, y, connect them with ℓ internally node-disjoint paths of length 2, and add new node z with edge yz. A longest induced path in this graph contains exactly 3 edges: yz and the two edges of one of the x-y-paths. By summing all constraints (2c) we can deduce

$$2|E| \geq \sum_{e \in E} \sum_{f \in \delta^*(e)} x_f \geq \underbrace{\sum_{e \in E} \sum_{f \in \delta^*(e) \cap E} x_f}_{a} + \underbrace{\sum_{v \in V} |\{e \in E : v \in e\}| \cdot x_{sv}}_{b}.$$

For the double sum a we see that any edge incident to x or z is considered ℓ times, while the other edges are considered $\ell + 1$ times. Thus $a \geq \ell \sum_{e \in E} x_e$. In the second sum b, yz is the only edge with coefficient 1 (instead of ≥ 2), and we thus have $b \geq 2 \sum_{v \in V} x_{sv} - x_{sz}$. By (2b) and the variable bound we have $b \geq 4 - 1 = 3$. Since $|E| = 2\ell + 1$ we overall have $2(2\ell + 1) \geq \ell \sum_{e \in E} x_e + 3$, giving objective value $\sum_{e \in E} x_e \leq 4 - \frac{1}{\ell}$. As the objective must be integral, this even yields the optimal bound 3 when using $\mathrm{LP}_{\mathrm{Base}}$ within an ILP solver.

We furthermore note that, to achieve strictly two-connected graphs, we could, e.g., also consider a cycle where each edge is replaced by two internally node-disjoint paths of length 2. However, in the above instance class the gap between the relaxations is larger, which is why we refrain from giving further details to the latter class. ◀

Since $\mathrm{ILP}_{\mathrm{Base}}$ is a sub-model of $\mathrm{ILP}_{\mathrm{Cut}}$ and $\mathrm{ILP}_{\mathrm{Flow}}$, this implies that the LP relaxations of the latter two are also stronger than $\mathrm{LP}_{\mathrm{Walk}}$. We note that by definition of the above programs and since $\mathrm{ILP}_{\mathrm{Base}}$ is not sufficient even for integral solutions, it is easy to see that $\mathrm{LP}_{\mathrm{Cut}}$ and $\mathrm{LP}_{\mathrm{Flow}}$ are in fact stronger than $\mathrm{LP}_{\mathrm{Base}}$. But we can also show:

▶ **Proposition 3.** *Let $\mathcal{P}_{\mathrm{Cut}}$ and $\mathcal{P}_{\mathrm{Flow}}$ be the polytope of* $\mathrm{LP}_{\mathrm{Cut}}$ *and* $\mathrm{LP}_{\mathrm{Flow}}$*, respectively. Let $\mathcal{P}'_{\mathrm{Flow}}$ be the projection of $\mathcal{P}_{\mathrm{Flow}}$ onto the x-variables by ignoring the z-variables. Then $\mathcal{P}_{\mathrm{Cut}} = \mathcal{P}'_{\mathrm{Flow}}$.*

Proof. We show that the projection is surjective. Clearly, it retains the objective value. We observe that by constraints (4a) for any node v there can be at most x_e units of flow along edge e that belong to a commodity of v. By constraints (4b,4c), each node $v \in V$ sends $\sum_{e \in \delta^*(v)} x_e$ units of flow that have to end in node s. Consequently, the claim—both that any $\mathrm{LP}_{\mathrm{Flow}}$ solution maps to an $\mathrm{LP}_{\mathrm{Cut}}$ solution and vice versa—follows directly from the duality of max-flow and min-cut. ◀

Recall that we may add clique constraints to either of these two models. Let $\mathrm{ILP}^k_{\mathrm{Cut}}$ and $\mathrm{ILP}^k_{\mathrm{Flow}}$ denote $\mathrm{ILP}_{\mathrm{Cut}}$ and $\mathrm{ILP}_{\mathrm{Flow}}$, respectively, augmented with all clique constraints for cliques on at most k nodes. From Proposition 3 we can deduce:

▶ **Corollary 4.** *Let $k \in \mathbb{N}$.* $\mathrm{LP}^k_{\mathrm{Cut}}$ *and* $\mathrm{LP}^k_{\mathrm{Flow}}$ *are equally strong.*

We show that increasing the clique sizes yields a hierarchy of ever stronger LP relaxations. Due to the above corollary, it suffices to consider the cut-based model in the following.

▶ **Proposition 5.** *For any $k \geq 4$, $\mathrm{LP}^{k}_{\mathrm{Cut}}$ is stronger than $\mathrm{LP}^{k-1}_{\mathrm{Cut}}$.*

Proof. $\mathrm{LP}^{k}_{\mathrm{Cut}}$ is as least as strong as $\mathrm{LP}^{k-1}_{\mathrm{Cut}}$ as we only add new constraints. Let $G = K_k$, the complete graph on k nodes. By choosing $Q = V$ in (5), $\mathrm{LP}^{k}_{\mathrm{Cut}}$ has objective value 1.

However, $\mathrm{LP}^{k-1}_{\mathrm{Cut}}$ allows a solution with objective value $\omega := 1 + \frac{2}{k-2} > 1$: We achieve this value by (fractionally) selecting all $\binom{k}{2}$ edges in G equally, i.e., we set $\tilde{x}_e := \omega / \binom{k}{2}$ for each $e \in E$. By additionally setting $\tilde{x}_{sv} = \frac{2}{k}$ for each $v \in V$, we obtain an LP feasible solution to $\mathrm{LP}^{k-1}_{\mathrm{Cut}}$: Clearly, constraints (2b,2c) are satisfied. The cut constraints (3a) are satisfied since edge variables are chosen uniformly (w.r.t. the two above edge types) and the right-hand side of the constraint sums over at least as many edge variables (per type) as the left-hand side. For any clique of size at most $k-1$, the left-hand side of its clique constraint (5) sums up to at most $\binom{k-1}{2} \cdot \tilde{x}_e = \binom{k-1}{2}(1 + \frac{2}{k-2}) / \binom{k}{2} = 1$.

We note that it is straight-forward to generalize G, so that it contains K_k only as a subgraph, while retaining the property of having a gap between the two considered LPs. ◀

5 Algorithmic Considerations

Separation. Since $\mathrm{ILP}_{\mathrm{Cut}}$ contains an exponential number of cut constraints (3a), it is not practical in its full form. We follow the traditional separation pattern for branch-and-cut-based ILP solvers: We initially omit cut constraints. Iteratively, given an LP solution feasible w.r.t. the some currently active constraints, we seek further cut constraints (that are in the model but not yet active) that this solution violates. In similar problems, adding cut constraints for singletons already initially was shown to be beneficial. However, in our case all singleton cut constraints (3a) are trivially always satisfied.

For a given LP solution, an edge $e \in E$ is called *active* if $x_e > 0$. A node is called active, if it has an incident active edge. These active graph elements yield a subgraph H of G^*.

For integral LP solutions, we simply compute the connected components of H and add a cut constraint for each component that does not contain s. Given a fractional LP solution $(\hat{x}, \hat{y})$, we compute the maximum flow value f_v between s and each active node v in H; the capacity of an edge $e \in E^*$ is equal to $\hat{x}_e$. If $f_v < \sum_{e \in \delta^*(v)} \hat{x}_e$, cut constraints based on the induced minimum s-v-cuts are added. Both routines manage to find a violated constraint if there is any. Note that already only separating on integral LP solutions suffices to obtain an exact algorithm—we simply may need more branching steps than with fractional separation.

Relaxing variables. As presented above, our models have $\Theta(|E|)$ binary variables, each of which may be used for branching by the ILP solver. We can reduce this number, by introducing $\Theta(|V|)$ new binary variables y_v, $v \in V$, that allow us to relax the binary x_e-variables, $e \in E$, to continuous variables. The new variables are precisely those discussed w.r.t. generalized subtour elimination, i.e., we require $y_v = \frac{1}{2} \sum_{e \in \delta^*(v)} x_e$. Assuming x_e to be continuous in $[0, 1]$, we have for every edge $e = \{v, w\} \in E$: if $y_v = 0$ or $y_w = 0$ then $x_e = 0$. Conversely, if $y_v = y_w = 1$ then $x_e = 1$ by (2c). Hence, requiring integrality for the y-variables (and, e.g., branching only on them), suffices to ensure integral x values.

Handling clique constraints. Since already finding a largest unweighted clique is NP-hard, we cannot hope for an exact separation routine for the clique constraints. Instead, we start with a heuristically found set of disjoint cliques in the graph, and add the corresponding constraints already in the initialization step.

6 Computational Experiments

Hard- and software. All algorithms are implemented in C++ using GNU gcc 8.3.0. We use Scip 6.0.1 [7] to implement the Branch-and-Cut algorithms with IBM Ilog Cplex 12.9.0 as the LP solver. To represent graphs in our implementations, we use the Open Graph Drawing Framework snapshot-2018-03-28 [4]. All tests are performed on an Intel Xeon Gold 6134 with 3.2 GHz and 256 GB RAM running Debian Stretch. Each test instance is limited to a single thread with a time limit of twenty minutes and a memory limit of 8 GB.

Considered algorithms. We re-implemented the strongest model from literature, ILP_{Walk}, including iteratively increasing the upper bound T to yield the best practical performance; we denote this algorithm by `W`. Since we use the same ILP solver for all considered algorithms, and given that the only implementation beyond standard generation of the ILP instance is a simple loop, we achieve a fair comparison between ILP_{Walk} and the new models.

For our implementation of ILP_{Cut} we consider various parameter settings w.r.t. to the algorithmic considerations as described in Section 5. We denote the arising algorithms chiefly by `C` to which we attach sub- and superscripts describing the parameters: the subscripts "int" and "frac" denote whether the separation is only done on integer, or also on fractional solutions. The superscript "n" specifies that we introduce node variables as the sole integer variables. Finally, the superscript "c" specifies that we use clique constraints. We consider all eight thereby possible ILP_{Cut} implementations.

As we see below, ILP_{Flow} performs clearly worse than any ILP_{Cut} variant in practice. We thus only report on the variant $\texttt{F}^{\text{n,c}}$; it is feature-wise analogous to the cut-based variant $\texttt{C}^{\text{n,c}}_{\text{frac}}$.

Considered instances. As a starting point, we consider the instance sets proposed for the problem in [13]. Overall, our test instances are grouped into four sets: `RWC`, `MG`, `BAS` and `BAL`.

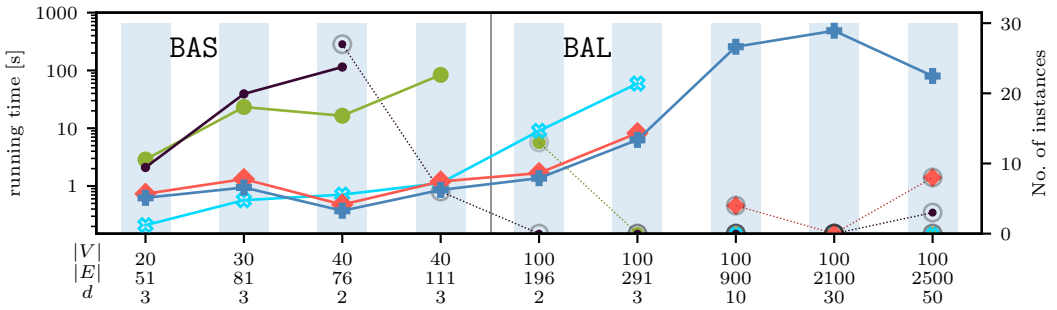
The first set, denoted `RWC`, is a collection of 22 real-world networks, including communication and social networks of companies and of characters in books, as well as transportation, biological, and technical networks. See [13] for details on the selection and `http://tcs.uos.de/research/lip` for the individual original sources. The *Movie Galaxy* (`MG`) set consists of 773 graphs representing social networks of movie characters [11]. While [13] considered only 17 of them, we use the full set here.

The other two sets are based on the Barabási-Albert probabilistic model for scale-free networks [1]. In [13], only the chosen parameter values are reported, not the actual instances. Our set `BAS` recreates instances with the same values: 30 graphs for each choice $(n, d) \in \{(20, 3), (30, 3), (40, 3), (40, 2)\}$, where $n = |V|$ and $d = \frac{|E|+1}{n}$ is the graph's density. As we will see, these small instances are rather easy for our models. We thus also consider a set `BAL` of graphs on 100 nodes; for each density $d \in \{2, 3, 10, 30, 50\}$ we generate 30 instances.

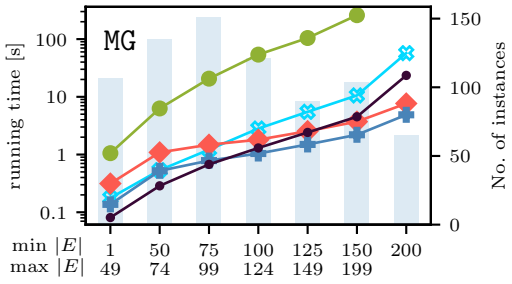
All instances and detailed results are available at `http://tcs.uos.de/research/lip`.

Comparing the implementations of the different formulations. We start with the most obvious question, whether the new formulations yield practically more effective implementations than the state-of-the-art. See Table 1 for `RWC`, Figure 1a for `BAS` and `BAL`, and Figure 1b for `MG`. For visual clarity, we restrict ourselves to three out of the possible eight implementations of ILP_{Cut} for now; we discuss the other strategies afterwards.

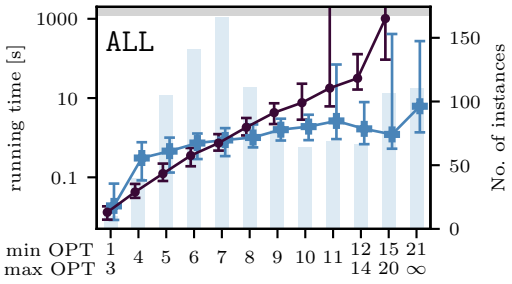
We observe that, rather independent on the benchmark set, the various ILP_{Cut} implementations achieve the best running times and success rates. The only exceptions are some small instances: there, the overhead of the stronger model, requiring an explicit separation



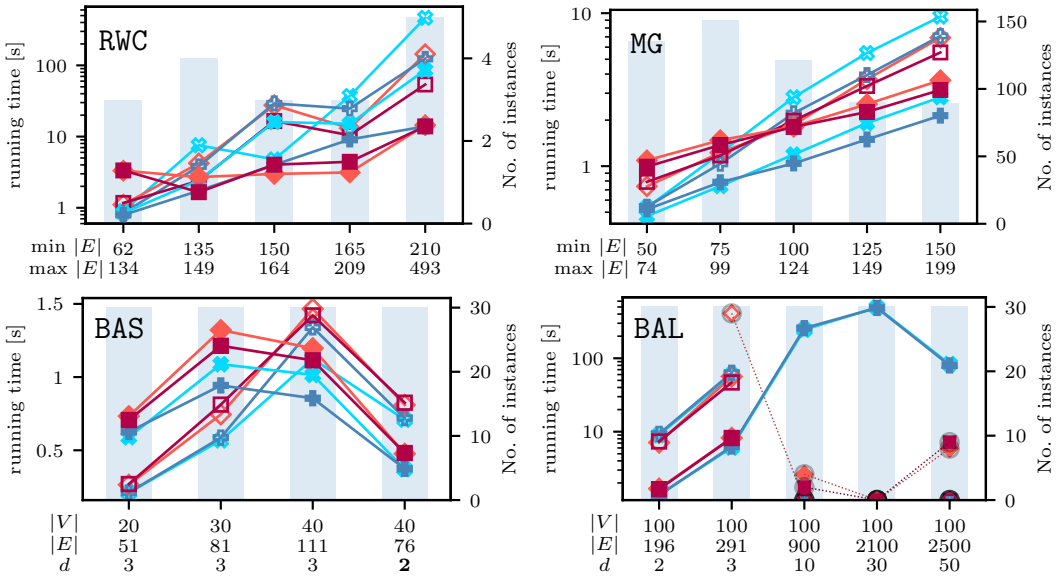
(a) Running time and success rate of ILP_{Walk}, ILP_{Flow}, and ILP_{Cut} implementations on `BAS` and `BAL`. The markers connected by solid lines give the median time (if below the timeout). Bars in the background give the number of instances. Gray encircled markers, connected via dotted lines, show the number of successful instances (if not 100%). See bottom of page for the legend.



(b) Running time of ILP_{Walk}, ILP_{Flow}, and ILP_{Cut} implementations on `MG`. The markers connected by solid lines give the median. Bars in the background give the number of instances.



(c) Running time vs. OPT over all instances. The solid lines give the median; for each data point we also give the 20% and 80% percentile. The gray area depicts the timeout.



(d) Median running times of different ILP_{Cut} implementations. For `RWC` and `MG`, we cluster the instances according to their size. Bars in the background give the number of instances. Gray encircled markers, connected via dotted lines, show the number of successful instances (if not 100%).

Figure 1 Comparisons between different ILP models.

Table 1 Running time (in seconds) on RWC for selected implementation variants. T and M denote failed computations due to time or memory limits, respectively. The best times are marked in bold.

instance	OPT	$\|V\|$	$\|E\|$	W	$\mathtt{F}^{\mathrm{n,c}}$	$\mathtt{C}_{\mathrm{int}}$	$\mathtt{C}_{\mathrm{frac}}$	$\mathtt{C}^{\mathrm{c}}_{\mathrm{int}}$	$\mathtt{C}^{\mathrm{c}}_{\mathrm{frac}}$	$\mathtt{C}^{\mathrm{n}}_{\mathrm{int}}$	$\mathtt{C}^{\mathrm{n}}_{\mathrm{frac}}$	$\mathtt{C}^{\mathrm{n,c}}_{\mathrm{int}}$	$\mathtt{C}^{\mathrm{n,c}}_{\mathrm{frac}}$
high-tech	13	33	91	15.40	45.15	0.90	1.11	0.84	1.17	0.51	0.81	**0.32**	2.76
karate	9	34	78	2.98	32.72	1.73	1.65	1.76	4.41	1.07	3.71	**0.79**	3.51
mexican	16	35	117	73.30	79.34	1.68	2.25	1.31	1.86	1.22	1.34	**1.04**	1.40
sawmill	18	36	62	70.00	6.53	0.51	**0.43**	0.51	0.45	0.85	3.32	0.84	3.35
tailorS1	13	39	158	83.80	199.38	4.78	7.92	4.66	6.76	1.51	1.87	**1.26**	2.03
chesapeake	16	39	170	106.00	166.18	**1.84**	13.11	2.43	10.43	2.29	4.88	3.26	4.42
tailorS2	15	39	223	445.00	432.92	6.80	21.78	16.26	17.77	**3.20**	4.31	3.27	4.89
attiro	31	59	128	T	219.89	1.76	2.57	2.56	1.82	1.20	1.75	**0.93**	1.23
krebs	17	62	153	522.00	470.83	**3.86**	28.21	29.23	16.52	16.00	11.26	6.19	4.03
dolphins	24	62	159	T	T	7.95	27.59	32.12	22.06	19.21	**2.99**	4.01	5.11
prison	36	67	142	T	567.21	13.36	5.87	5.07	2.93	3.62	4.05	**2.43**	1.91
huck	9	69	297	41.70	T	T	144.13	123.55	53.89	114.27	**11.63**	25.11	14.92
sanjuansur	38	75	144	T	T	30.67	8.64	28.08	10.56	8.22	**3.65**	5.04	3.84
jean	11	77	254	121.00	T	464.89	52.89	68.42	33.27	81.03	14.47	**7.75**	9.85
david	19	87	406	T	T	666.25	719.46	126.67	205.67	85.88	23.94	**13.91**	**13.91**
ieeebus	47	118	179	T	T	37.10	22.35	44.08	10.82	15.69	**3.13**	22.90	5.51
sfi	13	118	200	44.40	T	47.41	4.39	24.70	4.43	15.13	**2.64**	9.11	3.90
anna	20	138	493	T	T	21.58	296.69	219.20	T	439.23	20.27	**15.50**	16.71
usair	—	332	2126	T	M	T	T	T	T	T	T	T	T
494bus	142	494	586	T	T	T	379.29	T	386.63	T	178.92	T	**173.14**
662bus	—	662	906	T	M	T	T	T	T	T	T	T	T
yeast	—	2361	6646	T	M	T	T	T	T	T	T	T	T

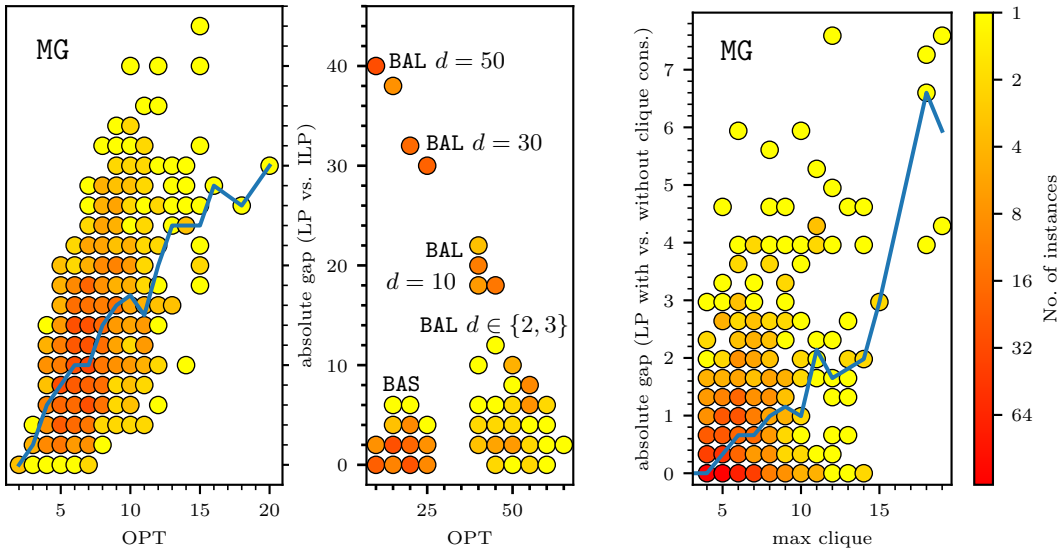
routine, does not pay off. Despite being formally equally strong as $\mathrm{ILP}_{\mathrm{Cut}}$, and stronger than $\mathrm{ILP}_{\mathrm{Walk}}$, the $\mathrm{ILP}_{\mathrm{Flow}}$ implementations cannot benefit from this in practice. This is consistent with known practical behavior of multi-commodity flow formulations: the benefit of being a compact (i.e., polynomially sized) model, is nullified by the practically much higher number of variables (compared to $\mathrm{ILP}_{\mathrm{Cut}}$) which typically becomes a bottleneck.

Only for MG, W yields comparable performance to the weaker of the cut-based variants; $\mathtt{F}^{\mathrm{n,c}}$ is weakest. On BAS instances, even $\mathtt{F}^{\mathrm{n,c}}$ is stronger than W (cf. Figure 1a), while the cut-based variants dominate: there even are variants (see below) that solve *all* of BAS. Since neither of these three instance sets (except for three exceptionally large instances in RWC) are particularly hard for the cut-based variants, we also consider larger random graphs, BAL: here, W and $\mathtt{F}^{\mathrm{n,c}}$ fail on virtually all instances. The cut-based model, however, allows certain implementations (see below for details) that solve all of these harder instances.

We want to point out one peculiarity on the BAL instances, visible in Figures 1a and 1d. The instances have 100 nodes but varying density. As the density increases from 2 to 30, the median running times of all algorithmic variants increase and the median success rates decrease. From $d = 30$ to $d = 50$, however, the running times drop again and the success rates increase. This is in fact to be expected: With $d = 50$, the graphs have more than half of all the possible edges. In dense graphs, the expected length of the maximum induced path is comparably low, and the instances become easier for all considered implementations.

Table 2 Counting how often a cut implementation was the fastest over all such implementations.

	$\mathtt{C}^{\mathrm{n,c}}_{\mathrm{int}}$	$\mathtt{C}^{\mathrm{n}}_{\mathrm{int}}$	$\mathtt{C}_{\mathrm{int}}$	$\mathtt{C}^{\mathrm{c}}_{\mathrm{int}}$	$\mathtt{C}_{\mathrm{frac}}$	$\mathtt{C}^{\mathrm{n}}_{\mathrm{frac}}$	$\mathtt{C}^{\mathrm{n,c}}_{\mathrm{frac}}$	$\mathtt{C}^{\mathrm{c}}_{\mathrm{frac}}$
all	428	296	78	77	51	49	43	42
MG	321	207	48	61	43	21	38	34
BAS,BAL	97	88	28	16	7	23	3	8
RWC	10	3	2	0	1	5	2	0



(a) LP value vs. OPT. Left: `MG`; right: `BAS` and `BAL`. Both plots share a common vertical axis.

(b) Maximal found clique size vs. LP value on `MG`.

Figure 2 Root LP relaxation value of the cut-based models. The blue lines show the median.

Comparing the different cut-based implementations. Choosing the best among the eight ILP_{Cut} implementations is not as clear as the general choice of ILP_{Cut} over ILP_{Flow} and ILP_{Walk}. We count the number of instances for which a particular implementation achieves the best running time, see Table 2. Even the weakest variant $\texttt{C}^{\text{c}}_{\text{frac}}$ still "wins" roughly 4% of the instances. A more in-depth inspection (see Figure 1d) shows that the two top configurations $\texttt{C}^{\text{n,c}}_{\text{int}}$ and $\texttt{C}^{\text{n}}_{\text{int}}$ behave very similarly throughout all instances. Whenever they are not the front-runners anyhow, the variants $\texttt{C}_{\text{int}}$ and $\texttt{C}^{\text{n}}_{\text{frac}}$ seem to be solid choices.

Generally, we can observe that the additional clique constraints never slow down the computation, but their benefit is rather minor. In contrast to this, adding additional node variables (and relaxing the integrality on the edge variables) nearly always pays off significantly. The probably most surprising finding is the choice of the separation routine: while the fractional variant is a quite fast algorithm and yields tighter dual bounds, the simpler integral separation performs better in practice. This is in stark contrast to seemingly similar scenarios like TSP or Steiner problems, where the former is considered by default. In our case, the latter—being very fast and called more rarely—is seemingly strong enough to find effective cutting planes that allow the ILP solver to achieve its computations fastest. This is particularly true when combined with the addition of node variables. In fact, $\texttt{C}^{\text{n,c}}_{\text{int}}$ and $\texttt{C}^{\text{n}}_{\text{int}}$ are the only two choices that can completely solve all large graphs in `BAL`.

Dependency of running time on the optimal value. Since the instance's optimal value OPT determines the final size of the ILP_{Walk} instance, it is natural to expect the running time of `W` to heavily depend on OPT. Figure 1c shows that this is indeed the case. The new models are less dependent on the solution size, as, e.g., witnessed by $\texttt{C}^{\text{n,c}}_{\text{int}}$ in the same figure.

Practical strength of the root relaxations. For our new models, we may ask how the integer optimal solution value and the value of the LP-relaxation (obtained by any cut-based implementation with exact fractional separation) differ, see Figure 2a. The gap increases for

larger values of OPT. Interestingly, we observe that the *density* of the instance seems to play an important role: for `BAS` and `BAL`, the plot shows obvious clusters, which—without a single exception—directly correspond to the different parameter settings as labeled. Denser graphs lead to weaker LP bounds in general. For `MG`, Figure 2b shows the relative improvement to the LP relaxation when adding clique constraints; the benefit increases when we are able to find larger cliques. Surprisingly, we neither observe any such benefit on `BAS` nor on `BAL`.

Conclusion. We propose two new ILP models for Longest Induced Path and prove that they both yield stronger relaxations in theory than the previous state-of-the-art. Moreover, we show that the cut-based model—generally, but also in particular in conjunction with further algorithmic considerations—clearly outperforms all other approaches in practice. We also provide strengthening inequalities based on cliques in the graph and prove that they form a hierarchy when increasing the size of the cliques.

Regarding the proposed clique inequalities it could be worthwhile to try to separate these inequalities (at least heuristically) to take advantage of their theoretical properties without overloading the initial model with too many such constraints. As it is unclear how to develop an *efficient* such separation scheme, we leave it as future research.

References

1. A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
2. P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. In *Proc. of STACS*, pages 256–268. Springer, 1989.
3. Y. Chen and J. Flum. On parameterized path and chordless path problems. In *Proc. of CCC*, pages 250–263. IEEE, 2007.
4. M. Chimani, C. Gutwenger, M. Juenger, G. W. Klau, K. Klein, and P. Mutzel. The Open Graph Drawing Framework (OGDF). In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization*, pages 543–569. Chapman and Hall/CRC, 2013. see `www.ogdf.net`.
5. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.
6. F. Gavril. Algorithms for maximum weight induced paths. *Inf.Proc.Let.*, 81(4):203–208, 2002.
7. A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, C. Schubert, F. Serrano, Y. Shinano, J. M. Viernickel, M. Walter, F. Wegscheider, J. T. Witt, and J. Witzig. The SCIP Optimization Suite 6.0. ZIB-Report 18-26, Zuse Institute Berlin, 2018. see `https://scip.zib.de`.
8. P. A. Golovach, D. Paulusma, and J. Song. Coloring graphs without short cycles and long induced paths. *Disc. Appl. Math.*, 167:107–120, 2014.
9. J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, 182(1):105–142, 1999.
10. L. Jaffke, O. Kwon, and J. A. Telle. Polynomial-time algorithms for the longest induced path and induced disjoint paths problems on graphs of bounded mim-width. In *Proc. of IPEC*, LIPIcs 89, pages 21:1–13, 2017.
11. J. Kaminski, M. Schober, R. Albaladejo, O. Zastupailo, and C. Hidalgo. Moviegalaxies - Social Networks in Movies. Harvard Dataverse, V3 2018. `doi:10.7910/DVN/T4HBA3`.
12. V. Lozin and D. Rautenbach. Some results on graphs without long induced paths. *Inf. Proc. Let.*, 88(4):167–171, 2003.
13. D. Matsypura, A. Veremyev, O. A. Prokopyev, and E. L. Pasiliao. On exact solution approaches for the longest induced path problem. *Euro. J. of Oper. Res.*, 278:546–562, 2019.
14. T. Uno and H. Satoh. An Efficient Algorithm for Enumerating Chordless Cycles and Chordless Paths. In *Proc. of Int. Conf. on Discov. Sci.*, LNCS 8777, pages 313–324, 2014.