Warm Starting Series of Mixed-Integer Linear Programs: Theory, Implementation, and Application

A Dissertation Proposal by Sean Kelley [a]
[a] Department of Industrial and Systems Engineering, Lehigh University

April 18, 2022

Abstract

Several well known classes of mixed-integer programs (MIP's) rely on solving series of mixed-integer linear programs (MILP's) where each instance differs only by objective coefficients or constraint bounds. Upon solving one MILP instance, attributes of its Branch-and-Cut tree can be reused to generate valid inequalities that tighten the underlying root linear programming (LP) relaxation for unsolved instances in the series. This approach could significantly improve how MILP's in such a series are warm-started today, but does not currently exist. This dissertation proposes the following deliverables to address that gap: publish a new warm-starting methodology with this approach, implement the methodology in open-source, and publish an application of the methodology to a sufficiently structured and common industrial application. This dissertation will yield emperical proof of how to more effectively warm start MILP's in a series and serve as a blueprint for improving solver performance for several classes of MIP's.

1 Introduction

1.1 Motivation

Mixed-Intger Programming (MIP) has a deeply studied theory and has been adopted in many successful industrial applications, such as production scheduling, vehicle routing, and facility location. Much of this theory and many of these applications are focused on the case in which a single Mixed-Integer Linear Programming (MILP) instance is solved. Driven by demand for applications to have more precise solutions, this theory has been expanded to enrich the space of solvable MIP instances. Some examples include solving MILP's with many constraints or variables by decomposition methods (electric grid planning and branch and price VRP), MILP's with multiple objectives (multiobjective facility location), MILP's with "real-time" run time restrictions (combinatorial auctions), and Bilevel MILP's (bilevel facility location). With this growth in theory, MIP is more relevant to industry than ever, specifically in the aforementioned applications.

A major bottleneck to implementing applications of more recently developed subclasses of MIP, like those mentioned above, is solving them in a reasonable amount of time. The computational difficultly in the above examples stems from the fact that each relies on solving a series of many MILP instances, formulated as follows:

$$\begin{aligned} &\text{minimize } {c^k}^T x \\ &\text{subject to } x \in \mathcal{S}^k \end{aligned} \tag{MILP-k}$$

where

$$\mathcal{S}^k = \{x : Ax \geq b^k,\ x_i \in \mathbb{Z} \text{ for all } i \in \mathcal{I}\}$$

In the above, x is our variable and $\mathcal{I}$ is the set of indices of x that must take on integer values. We assume $A \in \mathbb{R}^{m \times n}$, $b^k \in \mathbb{R}^m$, and $c^k \in \mathbb{R}^n$. For all indices k and l in the series, either $b^k = b^l$ or $c^k = c^l$.

It is worth noting that solving each instance of MILP-k is already an NP-hard problem. The burden when solving many in a series can be mitigated by "warm starting" each. Specifically, this could include using previously solved problems in the series to calculate what reasonable starting bounds, valid inequalities, or disjunction might be. This allows branch and cut in subsequent solves to bypass primal heuristics and/or initial exploration of the feasible region, and there exists two popular approaches today to accomplish this.

The most recently developed approach trains neural networks to predict primal heuristic and branching decisions made during Branch-and-Bound style algorithms [3], [2]. This enables solvers to more quickly navigate the branch and bound tree for new instances, even remaining effective for less rigidly structured series of instances due to the ability for neural networks to generalize. However, warm-starting with neural networks assumes the availability of the human and computational resources required to accurately train them. Given series of MILPs like MILP-k correspond largely to single problem instances within MIP, training a neural network for each one is intractible.

The traditional approach "reuses the Branch-and-Cut tree" from a previous solve, as is described in [7]. The primal and dual bounds in a series of MILPs with a fixed coefficient matrix are functions of the objective and constraint bounds. If one can recover those functions from a previously solved instance, they can be reevaluated at the objective and constraint bounds of the new instance to yield its starting bounds. Likewise, when fixing the coefficient matrix, many valid inequalities added during cut generation are functions of the constraint bounds [5]. These functions can be reevaluated at the constraint bounds of an unsolved instance to generate valid inequalities for it. One more benefit of a fixed coefficient matrix is that Branch-and-Cut for a new instance can be started from the terminal subproblems of a previous instance once their objective and constraint bounds have been updated. Doing so saves the solver from having to branch and bound repeatedly to generate those disjunctive terms. However, when the objective or constraint bounds differ significantly, the reevaluated valid inequalities and bounds can be weak, and Branch-and-Cut may process many subproblems from unneeded disjunctive terms [7]. Although the above warm starting techniques have little overhead, they can struggle to generalize well for series of MILP instances.

1.2 Outline

The demand for more precise solutions to the applications modeled by MIP's has led to richer, albeit more complex, solution methods, as mentioned in the previous section. For an important subset of these methods, their solves include a series of MILP's with MILP-k's structure. Given the structure each problem in such series shares with the others, there exists opportunity to warm-start new solves with information from previous ones, significantly improving the performance of the overall solve. However, given the gaps displayed in current warm-starting methodology, an approach that both generalizes well to MILP-k while requiring minimal effort from the practitioner is desired.

This dissertation proposes accomplishing the following to meet this need. First, a new warm starting methodology will be published. In that work, the methodology will show its ability to solve a series of MILP-k more efficiently than current warm starting methods. Second, this methodolgy will be implemented within a COIN-OR package, making this work accessible to the wider research community. Third, in a second publication, the methodology will be applied to a Branch-and-Price formulation of the Capactitated Vehichle Routing Problem with Time Windows, demonstrating its ability to outperform and solve a wider variety of instances than current implementations. Additionally, this publication will be a tutorial to the community of practitioners and researchers as to how they can integrate this warm-starting methodology within MIP algorithms and serve as evidence of this dissertation's effectiveness in practice. The end result will be publications and software that are ready to be built upon by future discrete optimizers.

2 Disjunctive Cut Warm Start Methodology

2.1 Inspiration

The first deliverable in the dissertation is to develop an improved methodology for warm-starting a series of MILP-k. As shown in section 1.1, training neural networks for individual MIP instances is cumbersome, so the proposed approach builds off of attributes from Branch-and-Cut trees from previously solved instances in the series. The inspiration is to translate the refined LP relaxation from a solved instance of MILP-k to an unsolved instance while removing all but one node from the queue when Branch-and-Cut begins on the unsolved instance. When compared to traditional warm-starting, this maintains the benefit of starting off with a tightened feasible region while removing the risk that many unneeded subproblems are solved should the objective or constraint bounds change sufficiently. To go deeper into the details of how this works, more notation is needed.

2.2 Preliminaries

Consider a Branch-and-Cut tree for MILP-k. Node t in this tree represents a LP subproblem solved during the course of the algorithm. This problem has the following formulation.

$$\begin{aligned} &\text{minimize } {c^k}^T x \\ &\text{subject to } x \in \mathcal{P}^{kt} \end{aligned} \tag{LP-kt}$$

where

$$\begin{aligned} \mathcal{P}^{kt} &= \{x : A^{tk} x \geq b^{tk},\ u^{kt} \geq x \geq l^{kt}\} \\ \mathcal{S}^{kt} &= \{x : x \in \mathcal{P}^{kt},\ x_i \in \mathbb{Z} \text{ for all } i \in \mathcal{I}\} \end{aligned}$$

$$A^{tk} = \begin{bmatrix} A \\ \Pi^{tk} \end{bmatrix} \text{ and } b^{tk} = \begin{bmatrix} b^k \\ \Pi_0^{tk} \end{bmatrix}$$

Above, u^{kt} and l^{kt} are bounds on x from branching on ancestor nodes. (Π^{kt}, Π_0^{kt}) represent constraints added to the LP relaxation from Branch-and-Cut's cut generation subroutine. Another important attribute of this tree is its set of terminal nodes, $\mathcal{T}^k$, which encodes the final disjunction on the variables.

A main goal of this publication is finding inequalities valid for all nodes in $\mathcal{T}^k$. For $\mu^{kt} \in \mathbb{R}^m_+$, $w^{kt} \in \mathbb{R}^n_+$, $v^{kt} \in \mathbb{R}^n_+$, and I_n the n-dimensional identity matrix, the following is a valid inequality for $\mathcal{P}^{kt}$:

$${A^{kt}}^T \mu^{kt} + I_n w^{kt} - I_n v^{kt} \geq {b^{kt}}^T \mu^{kt} + {l^{kt}}^T w^{kt} - {u^{kt}}^T v^{kt} \tag{LP-kt valid inequality}$$

Let $\pi \in \mathbb{R}^n$ and $\pi_0 \in \mathbb{R}$ satisfy the following:

$$\begin{aligned} \pi &\geq {A^{kt}}^T \mu^{kt} + I_n w^{kt} - I_n v^{kt} \\ \pi_0 &\leq {b^{kt}}^T \mu^{kt} + {l^{kt}}^T w^{kt} - {u^{kt}}^T v^{kt} \end{aligned} \quad \text{for all } t \in \mathcal{T}^k \tag{k-disjuctive cut}$$

By construction, k-disjuctive cut is a LP-kt valid inequality for all $t \in \mathcal{T}^k$.

Specifically, cuts that refine the underlying LP relaxation for MILP-k near its optimal solution are desired. For $\bar{x}\mathbb{R}^n$, an estimate of this solution, such a cut can be found by solving the Cut Generating LP for MILP-k,

which is defined as follows:

$$
\begin{aligned}
\text{minimize} \quad & \pi^T \bar{x} - \pi_0 \\
\text{subject to} \quad & \pi \geq {A^{kt}}^T \mu^{kt} + I_n w^{kt} - I_n v^{kt} && t \in \mathcal{T}^k \\
& \pi_0 \leq {b^{kt}}^T \mu^{kt} + {l^{kt}}^T w^{kt} - {u^{kt}}^T v^{kt} && t \in \mathcal{T} \\
& 1 = \sum_{t \in \mathcal{T}^k} \Big(\sum_{j=1}^{m^t} \mu_j^{kt} + \sum_{i=1}^{n} w_i^{kt} + \sum_{i=1}^{n} v_i^{kt} \Big) \\
& \mu^{kt} \in \mathbb{R}_+^{m^t},\ w^{kt} \in \mathbb{R}_+^n\ v^{kt} \in \mathbb{R}_+^n && t \in \mathcal{T}^k
\end{aligned}
\tag{CGLP-k}
$$

The added constraint from k-disjuctive cut to CGLP-k is for normalization to ensure the optimal solution is finite. In order to scale this LP to many disjunctive terms, a more performant variant [1] will be implemented and used for testing. Going forward, CGLP-k($\bar{x}'$) means to solve CGLP-k with $\bar{x} = \bar{x}'$. The above explains how to find k-disjuctive cuts for a solved MILP-k. The following shows a similar procedure can be used to find l-disjunctive cuts for an unsolved MILP-l.

In the case that $b^k = b^l$ and the same disjunction is applied, the cuts generated in each node t remain valid for $\mathcal{S}^{lt}$. Therefore, one can solve CGLP-k($\bar{x}^l$) to find l-disjunctive cuts.

In the case that $c^k = c^l$ and the same disjunction is applied, each LP-kt can be transformed to generate LP-lt such that it is a relaxation of $\mathcal{S}^{lt}$. Begin by replacing b^k with b^l. Then overload Π_0^{kt} to map b^l such that $(\Pi^{kt}, \Pi_0^{kt}(b^l))$ become valid inequalities for the integer points in LP-lt. For this publication, Gomory Mixed-Integer Cuts (GMICs) will be the only cuts used beyond those generated from the CGLP. See [5] for map details. Doing this yields the following constraints within $\mathcal{P}^{lt}$

$$
A^{lt} = \begin{bmatrix} A \\ \Pi^{kt} \end{bmatrix} \text{ and } b^{lt} = \begin{bmatrix} b^l \\ \Pi_0^{kt}(b^l) \end{bmatrix}
$$

Each of these terms can be substituted in CGLP-k to yield CGLP-l, which, given, $\bar{x}$, an estimate solution to MILP-l, is formulated as follows:

$$
\begin{aligned}
\text{minimize} \quad & \pi^T \bar{x} - \pi_0 \\
\text{subject to} \quad & \pi \geq {A^{lt}}^T \mu^{lt} + I_n w^{lt} - I_n v^{lt} && t \in \mathcal{T}^k \\
& \pi_0 \leq {b^{lt}}^T \mu^{lt} + {l^{lt}}^T w^{lt} - {u^{lt}}^T v^{lt} && t \in \mathcal{T}^k \\
& 1 = \sum_{t \in \mathcal{T}^k} \Big(\sum_{j=1}^{m^t} \mu_j^{lt} + \sum_{i=1}^{n} w_i^{lt} + \sum_{i=1}^{n} v_i^{lt} \Big) \\
& \mu^{lt} \in \mathbb{R}_+^{l^t},\ w^{lt} \in \mathbb{R}_+^n\ v^{lt} \in \mathbb{R}_+^n && t \in \mathcal{T}^k
\end{aligned}
\tag{CGLP-l}
$$

Analygous to CGLP-k, let CGLP-l($\bar{x}'$) represent solving CGLP-l with $\bar{x} = \bar{x}'$.

Warm starting MILP-l with disjunctive cuts means the following: Wtih a (partially) solved MILP-k, construct and run CGLP-k($\bar{x}'$) if $b^k = b^l$ or CGLP-l($\bar{x}'$) if $b^k \neq b^l$ for all $\bar{x}'$ solution estimates of MILP-l. Add the resulting cut(s) to the root relaxation of MILP-l and start Branch-and-Cut with the tightened root relaxation. Add further disjuctive cuts to the subtree rooted at a node t by solving CGLP-k($\bar{x}^{lt}$) or CGLP-l($\bar{x}^{lt}$) as appropriate where $\bar{x}^{lt}$ is the solution to LP-lt.

2.3 Numerical Experiments

There are three sets of experiments this publication will run, exploring each of the three possible cases for warm-starting MILP-l with disjunctive cuts: $c^l = c^k$ and $b^l = b^k$, $c^l = c^k$ but $b^l \neq b^k$, and $c^l \neq c^k$ but $b^l = b^k$. The procedure and deliverable for each experiment is as follows.

2.3.1 Restarting MILP-k

In the case that $c^l = c^k$ and $b^l = b^k$, MILP-l is the same as MILP-k. This scenario can explore if it is possible to partially solve MILP-k then warm-start with disjunctive cuts and solve to optimality MILP-l in less time than solving MILP-k to optimality alone. The first experiment of this publication will test to see which classes of MILP instances, if any, this happens. The deliverable will be a list of which MILP classes can be solved more efficiently in this manner.

Identifying such classes comes down to collecting and displaying a set of metrics while solving MILP-k and its disjunctive-cut warm-started peer, as initial runs will likely not yeild expected results and will lead to further refinements. To test if changing parameters affects the benefit of warm-starting, the following graphs are broken out by how many nodes are processed before restarting in the warm-start case, and whether or not variable bounds and the constraint matrix of CGLP-k accumulate as the solve progresses. When variable bounds are cumulative while solving CGLP-k at node t, for all nodes $t' \in \mathcal{T}^k$, $l^{kt} = \max\{l^{kt}, l^{kt'}\}$ and $u^{kt} = \min\{u^{kt}, u^{kt'}\}$. No updates to variable bounds are made in the case that they are fixed. When constraint bounds are cumulative while solving CGLP-k at node t, for all nodes $t' \in \mathcal{T}^k$, $\Pi^{t'k} = \Pi^{tk}$. Like the variable bounds, no updates to constraint matrices are made in the case that they are fixed. With the exception of figure 1, the graphs display a ratio of the warm and cold started cases. Thus, a positive number represents an improvement from warm-starting.

How often a node t fails to find a cut from CGLP-k that refines LP-kt is the first metric tracked and is displayed in figure 1. On average, 80% to 90% of nodes processed do not use the cut generated from CGLP-k. Since CGLP-k is constructed from 4 and 16 node disjunctions, this is an expected result, as the warm-started Branch and Bound tree does not need to be very deep before CGLP-k no longer generates cuts for a node t that refine LP-kt. This does, however, show that disjunctive cuts are used with a significant frequency, thus their effect may be felt by the performance of warm-started solves.

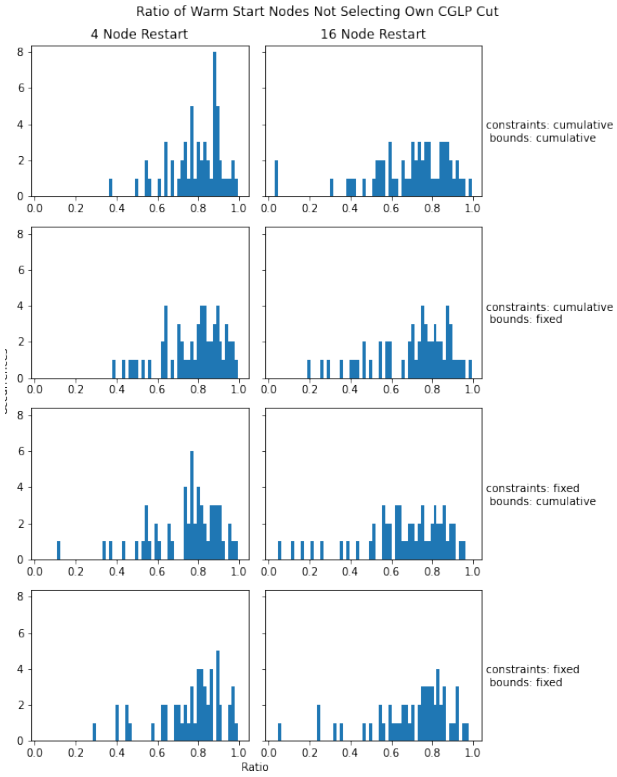

Figure 1: Ratio of nodes processed when warm-starting MILP-k that ignore the cut generated from CGLP-k

The first place to check for this performance improvement is in how much the initial dual bound improves after solving LP-k0, the root node relaxation for both the warm and cold started cases. Figure 2 illustrates this by plotting the ratio of the dual gaps after solving the root relaxations. The figure shows the extra cut added to LP-k0 does improve the dual bound by a couple percent on average, meaning that it is possible for the warm-started instances to see better overall solve performance.

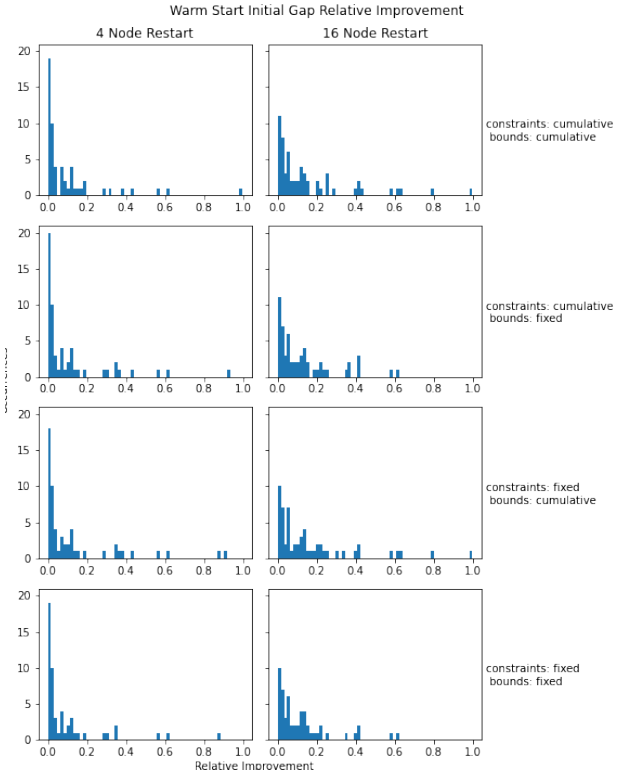

Figure 2: Ratio of dual gaps after solving the root relaxation for warm and cold started MILP-k instances

The first place to check if the overall solve is improved is the number of nodes both the warm and cold started MILP-k's had to process. The ratio of those two values is graphed in figure 3. This set of graphs reveals that in most cases adding cuts from CGLP-k leads to processing more nodes, which is the opposite of what is expected with tighter LP-kt's. To prove that warm-starting MILP-k is a viable approach to its solve, this dissertation will need to show that warm-starting reduces the number of nodes processed, which means it would then be plausible for warm-starting to reduce the run time compared to a cold start.

2.3.2 Warm Starting MILP-l When $c^l \neq c^k$ or $b^l \neq b^k$

Once shown that it is plausible to warm-start MILP-k more effectively than cold-starting, this dissertation will test the ability for disjuctive cuts from MILP-k to warm-start MILP-l. Similar to the data displayed in Section 2.3.1, both experiments in this section will be checked for how the root dual bound and total number of processed nodes changes between warm and cold starts.

Stepping up in complexity, the case that $c^l \neq c^k$ but $b^l = b^k$ will be tested next. This experiment will determine when warm-starting MILP-l with disjunctive cuts leads to a faster solve than the traditional warm-start approach and no warm-starting at all. In this experiment, since the feasible regions of MILP-k and MILP-l are the same, the initial solution estimates used for warm-starting could include all feasible solutions found to MILP-k. Additionally, the solution to the LP relaxation of MILP-k could be used. For determining when warm-starting MILP-l with disjunctive cuts has the best performance, various degrees of

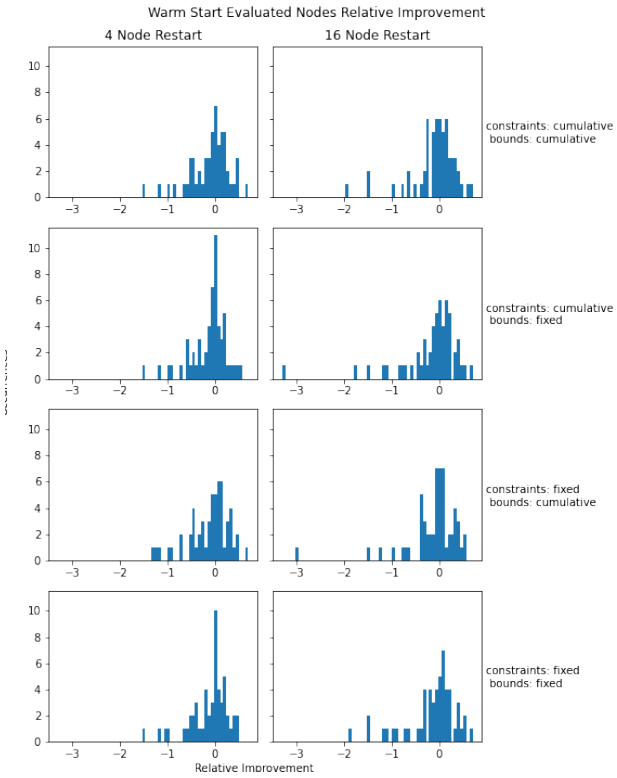

Figure 3: Ratio of number of nodes processed for warm and cold started MILP-k instances

change between c^l and c^k will be tested as will different classes of MILP-k. The deliverable is a range of classes of MILP-k and angles between c^l and c^k such that warm-starting MILP-l with disjunctive cuts leads to the most efficient solve.

Lastly, the case that $c^l = c^k$ but $b^l \neq b^k$ will be tested. This experiment will warm-start MILP-l when $b^k \neq b^l$ as proposed in the previous section. This experiment will proceed in the same way as the previous experiment and yield the same format of result.

2.4 Implications Moving Forward

In the near term, further work improving the algorithms that generate and select disjunctive cutting planes is needed. It is a reasonable expectation that warm-starting MILP-k with its own disjunctive cuts as in section 2.3.1 should see a reduction in number of nodes processed when solving. There could currently be a few reasons why this issue is occurring, and the current priority is to explore each to find potential improvements.

The first issue is that numerical errors could be hampering the accuracy of the cuts. In rare cases, the underlying optimization software that solves CGLP-k returns the wrong optimal solution. This could be happening more than is expected, which would reduce the quality of the cuts added to each LP-kt. To fix this, tests on smaller models that can be debugged need to be run to confirm CGLP-k is implemented correctly.

Another issue that comes to mind is that CGLP-k is not large enough to be effective. When the disjunction is only a few terms, CGLP-k is limited in the depth of cuts it can create as the disjunction is very similar to the root LP relaxation. Currently, it is hard to solve much larger instances of CGLP-k than tested in section 2.3.1, but [1] gives multiple ideas for how this computation can be improved. With those ideas implemented, future experiments can create better disjunctive cuts, possibly yielding more promising results.

One last issue to explore would be generating more than one cut from CGLP-k in each cut generation iteration. When CGLP-k is solved, it is relatively cheap computationally to find many nearby solutions, which are also valid inequalities for MILP-k. Augmenting the disjunctive cut generation subroutine to return many valid inequalities that could potentially be used to tighten LP-kt may help improve the overall solve performance by reducing the space needing searched for solutions.

Once the above changes are made and the three experiments are complete, there should ranges for classes of MILP's and degree of similarity between MILP-k and MILP-l that suggest which approach should be used to warm-start the latter given the former has already been partially solved. This information can be used to further improve the performance increase due to warm-starting classes of MIP reliant on solving series of MILP-k.

3 Implementing Disjunctive Cut Warm Starts

3.1 Motivation

Before disjunctive warm-starts can improve the performance of classes of MIP reliant on solving series of MILP-k, the code for disjunctive warm-starting must be open-sourced. Once open-sourced, the code will be available to the research community, enabling them to make those improvements. Further, another advantage of open-sourcing is that the code can integrate with a popular open-source solver (like CBC, SYMPHONY, or SCIP). The intention in this dissertation in doing the latter is to give legitimacy to experimental results found in the first publication. To implement disjunctive warm-starts in open-source, the following will need to be done.

3.2 Cut Generation Routine

First, a cut generation routine, which accepts a list of terminal subproblems to a solved MILP-k and a list of solution estimates to an unsolved MILP-l, must solve the appropriate CGLP and return the corresponding deepest cuts. For this to work, this dissertation will include adding a tree that tracks the generated subproblems for the augmented solver's Branch-and-Cut class if it does not already exist. Additionally, if there are subproblems that are optimized in a presolved space, then a subroutine will be developed to map preprocessed variables and constraint bounds back to the original problem space. With this information, the appropriate CGLP will be formulated and converted to the formulation of [1]. Then the found cuts will be returned.

For this to happen, all that remains is implementing the CGLP, and the approach depends on how MILP-k and MILP-l relate to one another. In the case that $b^k = b^l$, then, as explained in section 2.2, CGLP-k can be formulated directly from the Branch-and-Cut class's tree attribute. Another step or two will be required when $b^k \neq b^l$ to implement CGLP-l. In this case, all cuts added will need to either be removed or translated to remain valid. For the sake of balancing simplicity and effectiveness, only GMIC's will be considered for translation. Therefore, at a minimum, a subroutine to remove cuts generated will be developed. If it is a reasonable effort, attributes will be added to each subproblem tracking which cut types correspond to which rows of LP-kt and the parameters that generate each GMIC so they can be re-evaluated at b^l. After developing these subroutines and applying them to the Branch-and-Bound class's tree for MILP-k, the tree can build the CGLP in any case considered.

3.3 Warm Start Class

Beyond a cut generation method, the other piece to implement is a warm-starting class, which has one defining attribute and method each. The attribute of this class is a list of MILP-k represented as Branch-and-Bound objects. The method accepts a MILP-l and a specification to warm-start with MILP-k such that $b^k = b^l$ or $c^k = c^l$. It then calls the new cut generation routine using the Branch-and-Bound object from the nearest sufficient MILP-k and the solution to LP-$l0$, the LP relaxation of MILP-l, adding the resulting cut to

LP-$l0$. (Nearest in this case means most parallel objectives if $b^k = b^l$ or smallest eucidian distance between constraint bounds if $c^k = c^l$.) The method finishes by running the solve routine on MILP-l's Branch-and-Bound instance. Optionally, this method could also invoke Branch-and-Bound to include CGLP in its cut generation routine to possibly further improve performance. Lastly, the class instantiates an object with a Branch-and-Bound instance it then solves and places in its attribute's list.

3.4 Implications Moving Forward

With an open-sourced implementation, the first publication of this dissertation can guide future research efforts as to how classes of MIP's solving series of MILP-k should be warm started.

4 Applying Disjunctive Cut Warm Starts

4.1 Motivation

As mentioned earlier, the objective with this dissertation is to advance understanding of how to warm start series of MILP-k's and ensure its incorporation into future research. The former will be accomplished by the above publication and open-source contribution. The second publication of this disseration will accomplish the latter by providing context on the effectiveness and ease of disjunctive cut warm starting in practice. Specifically, the second publication will solve the Capacitated Vehicle Routing Problem with Time Windows (CVRPRTW) via Branch-and-Price where the pricing problem is solved as a series of MILP-k's warm started with disjunctive cuts. This dissertation picks this problem and approach for the following reasons.

Branch-and-Price is one of the most commonly applied methods that solves a series of MILP-k, and the CVRPTW is one of the most impactful problems solved with that approach. Thus, showing performance improvements in solving this problem would have immediate impact for how this problem is solved in practice. This kind of accomplishment would generate additional attention for this dissertation's first publication and open-source software contribution, greatly improving the likelihood the research community builds upon those works.

Further, this publication explores warm-starting the CVRPTW with Branch-and-Price because it can be improved along multiple frontiers. Current approaches to the CVRPTW with Branch-and-Price use the pricing problem to generate routes and the master problem to select the optimal subset of routes generated that ensure demand is met [4]. In such approaches, route generation is relaxed so Dynamic Programming can solve it, as using MILP is thought to be too inefficient.

This presents two opportunities. First, warm-starting the pricing problem with disjunctive cuts will challenge the notion that Dynamic Programming (DP) is more efficient. The pricing problem is solved many times, increasing the effectiveness of warm-starts, possibly to the extent that solving MILP's is faster than DP's. Second, the DP relaxation of the pricing problem removes the ability for practitioners to add on "side-constraints" that ensure CVRPTW respect real-world restrictions. As [6] shows, side-constraints are a necessary feature of industrial CVRPTW implementations, which is keeping the benefits of Branch-and-Price from being realized at the highest level of impact. Since warm-starting with disjunctive cuts enables solving the full pricing problem efficiently, this publication promises to change that.

Ancillary motivations exist because the CVRPTW is well studied. For one, there exists a rich sample set of instances to test against. As a result, this publication will show how broadly it is effective against current Branch-and-Price implementations, lending further credibility to the results. Further, Branch-and-Price approaches to CVRPTW are well known, allowing the changes in implementation to incorporate disjunctive cut warm-starts to stand out. The result here is an explicit tutorial on how to apply this dissertation's methodology so it can be applied elsewhere.

4.2 Numerical Experiments

The above section largely spells out what this publication intends to test. The recap is as follows. Both experiments will solve an array of instances of the CVRPTW with Branch-and-Price. The following is how they differ.

In the first experiment, instances will be restricted to those whose pricing problem can be solved with DP. Each instance will have its pricing problem solved as a DP then as a MILP-k warm-started with disjunctive cuts. This will show over what range of instances warm-starting with disjunctive cuts is more efficient than solving a DP.

In the second experiment, instances with challenging pricing problems that cannot be solved with DP will be chosen. This experiment will compare warm-starting the pricing problem to cold starts and heuristic methods to profile how all three approaches differ in terms of solution quality and efficiency. The deliverable is to show under what circumstances pricing problems that cannot be solved with DP's should be warm-started and how impactful is such an approach as compared to cold-starting.

4.3 Implications of Applying Disjunctive Cut Warm-Starts

The implications of applying the first two parts of this dissertation are the motivation, which is mentioned above. To recap, disjunctive cut warm-starts should be applied to Branch-and-Price approaches to the CVRPTW because it is an impactful application in which this dissertation's previous contributions will be effective. Additionally, since Branch-and-Price is well known, this publication will serve as an effective tutorial as to how future publications can apply the earlier works of this dissertation. Collectively, those two results will highlight the utility of this dissertation and prototype future research directions.

5 Conclusion

In conclusion, this dissertation will develop a new warm-starting methodology, implementation, and application by leveraging disjunctive cuts. It contributes a novel idea to the discrete optimization community with clear directions on how and where to build upon it in the future. Given the breadth of research activity required and that much of the work has and will be done remotely, this dissertation ensures my development into an independent researcher, the main goal of the Ph.D. program. For these reasons, I propose this research plan for my dissertation.

References

[1] Balas and Kazachov. V-polyhedral disjunctive cuts. *N/A*, 2021.

[2] Lodi et. al. Learning to schedule heuristics in branch-and-bound. *ArXiv*, 2019. doi:https://arxiv.org/abs/2103.10294.

[3] Nair et. al. Solving mixed integer programs using neural networks. *ArXiv*, 2020. doi:https://arxiv.org/pdf/math/0411298v1.pdf.

[4] Feillet. A tutorial on column generation and branch-and-price for vehicle routing problems. *Springer*, 2010.

[5] Ralphs Guzelsoy. Duality for mixed-integer linear programs. *COR@L*, 2007. doi:https://www.researchgate.net/publication/228383307_Duality_for_mixed-integer_linear_programs.

[6] et. al. Khodabandeh. Ch robinson uses heuristics to solve rich vehicle routing problems. *INFORMS Journal on Applied Analytics*, 2021.

[7] Ralphs. Warm starting mixed integer programs. *COR@L*, 2006. doi:https://coral.ise.lehigh.edu/~ted/files/papers/DMII06.pdf.