Surrogate "Level-Based" Lagrangian Relaxation for Mixed-Integer Linear Programming

Mikhail Bragin (mbragin@engr.ucr.edu)
University of California, Riverside

Emily Tucker
Clemson University

Article

Keywords: Mixed-Integer Linear Programming, Combinatorial Optimization, Discrete Optimization, Lagrangian Relaxation, Decomposition and Coordination, Generalized Assignment Problems, Manufacturing Scheduling, Pharmaceutical Scheduling

Posted Date: September 26th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-2011036/v2

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
Surrogate “Level-Based” Lagrangian Relaxation for Mixed-Integer Linear Programming

Mikhail A. Bragin\textsuperscript{1} and Emily L. Tucker\textsuperscript{2}

\textsuperscript{1}Department of Electrical and Computer Engineering, University of Connecticut, 371 Fairfield Way, Unit 4157, Storrs, 06269, CT, USA.

\textsuperscript{2}Department of Industrial Engineering, Clemson University, 271 Freeman Hall, Clemson, SC, 29634, USA.

Contributing authors: mikhail.bragin@uconn.edu; mikhail.bragin@ucr.edu; etucke3@clemson.edu;

Abstract

Mixed-Integer Linear Programming (MILP) plays an important role across a range of scientific disciplines and within areas of strategic importance to society. The MILP problems, however, suffer from combinatorial complexity. Because of integer decision variables, as the problem size increases, the number of possible solutions increases super-linearly thereby leading to a drastic increase in the computational effort. To efficiently solve MILP problems, a “price-based” decomposition and coordination approach is developed to exploit 1. the super-linear reduction of complexity upon the decomposition and 2. the geometric convergence potential inherent to Polyak’s step-sizing formula for the fastest coordination possible to obtain near-optimal solutions in a computationally efficient manner. Unlike all previous methods to set step-sizes heuristically by adjusting hyper-parameters, the key novel way to obtain step-sizes is purely decision-based: a novel “auxiliary” constraint satisfaction problem is solved, from which the appropriate step-sizes are inferred. Testing results for large-scale Generalized Assignment Problems (GAP) demonstrate that for the majority of instances, certifiably optimal solutions are obtained. For stochastic job-shop scheduling as well as for pharmaceutical scheduling, computational results demonstrate the two orders of magnitude speedup
as compared to frequently used Branch-and-Cut (B&C). SLBLR has a major impact on the efficient resolution of complex Mixed-Integer Programming (MIP) problems arising within a variety of scientific fields.

**Keywords:** Mixed-Integer Linear Programming; Combinatorial Optimization; Discrete Optimization; Lagrangian Relaxation; Decomposition and Coordination; Generalized Assignment Problems; Manufacturing Scheduling; Pharmaceutical Scheduling

### 1 Introduction

Mixed-Integer Linear Programming (MILP) plays an important role across a range of scientific disciplines such as mathematics, operations research, engineering, computer science as well as within a range of areas of strategic importance to society such as manufacturing [1], pharmacy [2–5], healthcare [6, 7], humanitarian applications [8], biology [9, 10], power and energy systems [11], and many others. The associated systems are created by interconnecting smaller subsystems, each having its own objective and a set of constraints. The subsystem interconnection is modeled through the use of *system-wide coupling* constraints. Accordingly, the MILP problems are frequently formulated in terms of cost components associated with each subsystem with the corresponding objective functions being additive as such:

\[
\min_{(x,y):=(x_i,y_i)_{i=1}} \left\{ \sum_{i=1}^{I} \left( c^x_i x_i + c^y_i y_i \right) \right\}.
\]

Furthermore, coupling constraints are additive in terms of subsystems:

\[
s.t. \sum_{i=1}^{I} A^x_i x_i + \sum_{i=1}^{I} A^y_i y_i - b = 0, \quad \{x_i, y_i\} \in F_i, i = 1, \ldots, I.
\]

The *primal* problem (1)-(2) is assumed to be feasible and the feasible region \( F_i \subset \mathbb{Z}_{n_i} \times \mathbb{R}_{m_i} \) is assumed to be bounded and finite. The MILP problems modeling the above systems are referred to as *separable*. Because of the discrete decisions, however, MILP problems are known to be NP-hard and are prone to the curse of *combinatorial complexity*. As the size of a problem increases, the associated number of combinations of possible solutions (hence the term “combinatorial”) increases super-linearly (e.g., exponentially) thereby making problems of practical sizes difficult to solve to optimality; even near-optimal solutions are frequently difficult to obtain.

A beacon of hope to resolve combinatorial difficulties lies through the exploitation of separability through the *dual* “price-based” decomposition and coordination Lagrangian Relaxation technique. After relaxation of coupling constraints (2), the coordination of subproblems amounts to the maximization of a concave non-smooth dual function:
\begin{align*}
\max_\lambda \{ q(\lambda) : \lambda \in \mathbb{R}^m \} , \\
\text{where} \\
q(\lambda) &= \min_{(x,y)} \left\{ L(x,y,\lambda), \{ x_i, y_i \} \in \mathcal{F}_i, i = 1, \ldots, I \right\}.
\end{align*}

Here \( L(x,y,\lambda) \equiv \sum_{i=1}^{I} (c^x_i)^T x_i + \sum_{i=1}^{I} (c^y_i)^T y_i + \lambda^T \cdot \left( \sum_{i=1}^{I} A^x_i x_i + \sum_{i=1}^{I} A^y_i y_i - b \right) \) is the Lagrangian function. The Lagrangian multipliers \( \lambda \) (\"dual\" variables) are the decision variables with respect to the dual problem (3), and it is assumed that the set of optimal solutions is not empty. The minimization within (4) with respect to \( \{ x, y \} \) is referred to as the \"relaxed problem.\"

The \textit{bright side} of the NP-hardness is the \textit{exponential} reduction of complexity upon the reduction of the problem size upon the decomposition. The decomposition \"reverses\" the combinatorial complexity thereby exponentially reducing the effort required to solve the resulting subproblems:

\[ \min_{x_i, y_i} \left\{ (c^x_i)^T x_i + (c^y_i)^T y_i + \lambda^T \cdot (A^x_i x_i + A^y_i y_i) \right\} , \]

thereby offering a viable potential to improve operations of existing systems as well as to scale up the size of the systems to support their efficient operations.

While decomposition efficiently reduces the combinatorial complexity, the coordination aspect of the method to efficiently obtain the optimal \"prices\" (Lagrangian multipliers) has been the subject of an intense research debate for decades because of fundamental difficulties of the non-smooth optimization. Namely, because of the presence of integer variables \( x \), the dual function (3) is non-smooth comprised of flat convex polygonal facets (each corresponding to a particular solution to the relaxed problem within (4)) intersecting at linear ridges along which the dual function \( q(\lambda) \) is non-differentiable; in particular, \( q(\lambda) \) is not differentiable at \( \lambda^* \) thereby ruling out the possibility of using necessary and sufficient conditions for the extremum. As a result of non-differentiability of \( q(\lambda) \), subgradient multiplier-updating directions, however, are non-ascending directions thereby leading to a decrease of dual values; subgradient directions may also change drastically thereby resulting in zigzagging of Lagrangian multipliers (see Figure 1 for illustrations) and slow convergence as a result.

Traditional methods to maximize \( q(\lambda) \) rely upon iterative updates of Lagrangian multipliers by taking a series of steps \( s^k \) along subgradient \( g(x^k, y^k) \) directions as:

\[ \lambda^{k+1} = \lambda^k + s^k \cdot g(x^k, y^k) , \]

where \( \{ x_i^k, y_i^k \} \equiv \{ x_i^k, y_i^k \}_{i=1}^{I} \) is an optimal solution to the relaxed problem (4) with multipliers equal to \( \lambda^k \). Within the Lagrangian Relaxation framework, subgradients are defined as levels of constraint violations \( g(x^k, y^k) \equiv \ldots \)
Fig. 1 An example of a dual function demonstrating difficulties faced by subgradient methods. Solid lines denote the level curves, dash-dotted lines denote the ridges of the dual function whereby the usual gradients are not defined (possible subgradient directions at points A and B are denoted by solid arrows), and the direction from point B toward optimal multipliers is denoted by a dashed line.

$$\sum_{i=1}^{I} A_i^x x_i^k + \sum_{i=1}^{I} A_i^y y_i^k - b.$$  Inequality constraints $\sum_{i=1}^{I} A_i^x x_i + \sum_{i=1}^{I} A_i^y y_i \leq b$, if present, can be handled by converting into equality constraints by introducing non-negative real-valued slack variables $z$ such that $\sum_{i=1}^{I} A_i^x x_i + \sum_{i=1}^{I} A_i^y y_i + z = b$.

Because of the lack of differentiability of $q(\lambda)$, notably, at the optimum $\lambda^*$, the step-size selection plays an important role to guarantee convergence to the optimum as well as for the success of the overall Lagrangian Relaxation methodology for solving MILP problems.

One of the earlier papers on the optimization of non-smooth convex functions, with $q(\lambda)$ being its member, though irrespective of Lagrangian Relaxation, is Polyak’s seminal work [12]. Intending to achieve geometric (also referred to as “linear”) rate of convergence so that $\|\lambda^k - \lambda^*\|$ is monotonically decreasing, Polyak proposed the stepsizing formula, which in terms of the problem under consideration takes the following form:

$$0 < s^k < \gamma \cdot \frac{q(\lambda^*) - q(\lambda^k)}{\|g(x^k, y^k)\|^2}, \gamma < 2. \quad (7)$$

Within (7) and thereafter unless stated otherwise, the standard Euclidean norm is used.

Subgradient directions, however, 1. are generally difficult to obtain computationally when the number of subproblems (5) to be solved is large, 2. change drastically thereby resulting in zigzagging of Lagrangian multipliers and slow convergence, and 3. the lack of the knowledge about the optimal dual value $q(\lambda^*)$ to set stepsizes.

To overcome the first two of the difficulties above, the Surrogate Subgradient Method was developed by [13] whereby the exact optimality of the relaxed problem (or even subproblems) is not required. As long as the following
“surrogate optimality condition” is satisfied:

\[ L(\tilde{x}^k, \tilde{y}^k, \lambda^k) < L(\tilde{x}^{k-1}, \tilde{y}^{k-1}, \lambda^k) \]  \hspace{1cm} (8)

the multipliers can be updated by using the following formula

\[ 0 < s^k < \gamma \cdot \frac{q(\lambda^*) - L(\tilde{x}^k, \tilde{y}^k, \lambda^k)}{\|g(\tilde{x}^k, \tilde{y}^k)\|^2}, \quad \gamma < 1, \]  \hspace{1cm} (9)

and convergence to \( \lambda^* \) is guaranteed. Here “tilde” is used to distinguish solutions \( \{x^k\} \), which are obtained by optimally solving the relaxed problem, from the solutions of the relaxed problem \( \{\tilde{x}^k\} \), not necessarily optimal, but satisfying the “surrogate optimality condition” (8). Unlike that in Polyak’s formula, parameter \( \gamma \) is less than 1 to guarantee that \( q(\lambda^*) > L(\tilde{x}^k, \tilde{y}^k, \lambda^k) \) so that the step-sizing formula (9) is well-defined in the first place, as proved in [13, Proposition 3.1, p. 703]. Once \( \{\tilde{x}^k, \tilde{y}^k\} \) are obtained, multipliers are updated by using the same formula as in (6) with stepsizes from (9) and “surrogate subgradient” multiplier-updating directions \( g(\tilde{x}^k, \tilde{y}^k) \) in place of subgradient directions \( g(x^k, y^k) \). Since only one subproblem is solved at a time, “surrogate” multiplier-updating directions do not change drastically and the concomitant reduction of multiplier zigzagging has also been observed. The main difficulty of the method is that the knowledge about \( q(\lambda^*) \) is unavailable. As a result, geometric/linear convergence of the method (or any convergence at all) is highly questionable. Nevertheless, the underlying geometric convergence principle behind the formula (8) is promising and will be exploited in Section 2.

Because of the presence of the integer variables, there is the so-called the duality gap, which means that even at convergence, \( q(\lambda^*) \) is generally less than the optimal cost of the original problem (1)-(2). To obtain a feasible solution to (1)-(2), the subproblem solutions when put together may not satisfy all the relaxed constraints. Therefore, to solve corresponding MILP problems, heuristics are inevitable and are used to perturb subproblem solutions. The important remark here is that the closer the multipliers are to the optimum, generally, the closer the subproblem solutions are to the global optimum of the original problem, and the easier it is to obtain feasible solutions through heuristics. Therefore, having fast convergence in the dual space to maximize the dual function (3) is of paramount importance for the overall success of the method. Specific heuristics will be discussed in subsection 2.2 and it will be demonstrated that for multiple difficult cases tested, the optimal solution to (1)-(2) is recovered though heuristics.

2 Results

2.1 Surrogate “Level-Based” Lagrangian Relaxation

In this subsection, a novel Surrogate “Level-Based” Lagrangian Relaxation (SLBLR) method is developed with adaptive adjustments of “level” estimates.
of $q(\lambda^*)$ within the Polyak’s step-sizing formula for fast convergence of multipliers when optimizing the dual function (3). Since the knowledge of $q(\lambda^*)$ is generally unavailable, over-estimates of the optimal dual value, if used in place of $q(\lambda^*)$ within the formula (9), may lead to oscillation of multipliers and to the divergence. Rather than using heuristic “oscillation detection” of multipliers used to adjust “level” values [14], the key of SLBLR is the decision-based “divergence detection” of multipliers based on a novel auxiliary “multiplier-divergence-detection” constraint satisfaction problem.

“Multiplier-Divergence-Detection” Problem to Obtain the Estimate of $q(\lambda^*)$. The premise behind the multiplier-divergence detection is the rendition of the result due [13, Theorem 4.1, p. 706]:

**Theorem 1** Under the stepsizing formula

$$s^k < \gamma \cdot \frac{q(\lambda^*) - L(\tilde{x}^k, \tilde{y}^k, \lambda^k)}{\|g(\tilde{x}^k, \tilde{y}^k)\|^2}, \gamma < 1,$$

(10)

such that \{\tilde{x}^k, \tilde{y}^k\} satisfy

$$L(\tilde{x}^k, \tilde{y}^k, \lambda^k) \leq L(\tilde{x}^{k-1}, \tilde{y}^{k-1}, \lambda^k),$$

(11)

the multipliers move closer to optimal multipliers $\lambda^*$ iteration by iteration:

$$\|\lambda^* - \lambda^{k+1}\| < \|\lambda^* - \lambda^k\|.$$  

(12)

The following two theorems are the main key results of this paper.

**Theorem 2** If

$$\|\lambda^* - \lambda^{k+1}\| \geq \|\lambda^* - \lambda^k\|,$$

(13)

then

$$s^k \geq \gamma \cdot \frac{q(\lambda^*) - L(\tilde{x}^k, \tilde{y}^k, \lambda^k)}{\|g(\tilde{x}^k, \tilde{y}^k)\|^2}.$$  

(14)

**Proof** Proof of Theorem 2. Define two predicates

$$A = \left\{ s^k < \gamma \cdot \frac{q(\lambda^*) - L(\tilde{x}^k, \tilde{y}^k, \lambda^k)}{\|g(\tilde{x}^k, \tilde{y}^k)\|^2} \right\},$$

(15)

and

$$B = \left\{ \|\lambda^* - \lambda^{k+1}\| < \|\lambda^* - \lambda^k\| \right\}.$$  

(16)

From Theorem 1 the following is true $A \Rightarrow B$ and from the Corollary 1 the following is true $\sim B \Rightarrow \sim A$. It remains to prove that both assertions are equivalent. Taking negation of $A \Rightarrow B$ leads to $\sim A \lor B$, and taking negation of $\sim B \Rightarrow \sim A$ leads to $\sim (\sim B) \lor \sim A$, which simplifies to $B \lor \sim A$. □
Theorem 3 If the following auxiliary “multiplier-divergence-detection” feasibility problem (with $\lambda$ being a continuous decision variable: $\lambda \in \mathbb{R}^m$)

$$
\begin{cases}
\|\lambda - \lambda^{k_j+1}\| \leq \|\lambda - \lambda^{k_j}\|,
\|\lambda - \lambda^{k_j+2}\| \leq \|\lambda - \lambda^{k_j+1}\|,
\vdots
\|\lambda - \lambda^{k_j+n_j}\| \leq \|\lambda - \lambda^{k_j+n_j-1}\|,
\end{cases}
$$

(17)

admits no feasible solution with respect to $\lambda$ for some $k_j$ and $n_j$, then $\exists \kappa \in [k_j, k_j + n_j]$ such that

$$
s^\kappa \geq \gamma \cdot \frac{q(\lambda^\kappa) - L(\tilde{x}^\kappa, \tilde{y}^\kappa, \lambda^\kappa)}{\|g(\tilde{x}^\kappa, \tilde{y}^\kappa)\|^2}.
$$

(18)

Proof Proof of Theorem 3. Assume the contrary: $\forall \kappa \in [k_j, k_j + n_j]$ the following holds:

$$
s^\kappa < \gamma \cdot \frac{q(\lambda^\kappa) - L(\tilde{x}^\kappa, \tilde{y}^\kappa, \lambda^\kappa)}{\|g(\tilde{x}^\kappa, \tilde{y}^\kappa)\|^2}.
$$

(19)

By Theorem 1, multipliers approach $\lambda^*$, therefore, the “multiplier-divergence-detection” problem admits at least one feasible solution - $\lambda^*$. Contradiction. □

From (19) it follows that $\exists \overline{q}_{k_j}$ such that $\overline{q}_{k_j} > q(\lambda^*)$ and the following holds:

$$
\overline{q}_{k_j} = \gamma \cdot \frac{\overline{q}_{k_j} - L(\tilde{x}^\kappa, \tilde{y}^\kappa, \lambda^\kappa)}{\|g(\tilde{x}^\kappa, \tilde{y}^\kappa)\|^2}.
$$

(20)

The equation (20) can equivalently be rewritten as:

$$
\overline{q}_{k_j} = \frac{1}{\gamma} \cdot s^1 \cdot \|g(\tilde{x}^\kappa, \tilde{y}^\kappa)\|^2 + L(\tilde{x}^\kappa, \tilde{y}^\kappa, \lambda^\kappa).
$$

(21)

Therefore,

$$
\overline{q}_j = \max_{\kappa \in [k_j, k_j + n_j]} \overline{q}_{k_j} > q(\lambda^*).
$$

(22)

A brief yet important discussion is in order here. The overestimate $\overline{q}_j$ of the dual value $q(\lambda^*)$ is the sought-for “level” value after the $j^{th}$ update (the $j^{th}$ time the problem (17) is infeasible). Unlike previous methods, which require hyper-parameters to obtain “level” values, up to this point, SLBLR is hyper-parameter-free. Specifically, neither “multiplier-divergence-detection” problem (17), nor the computations within (20)-(22) requires hyper-parameter adjustment; following [15], parameter $\gamma$ will be chosen as a fixed value $\gamma = \frac{1}{2}$, which is the inverse of the number of sub-problems, and will not require further adjustments. Moreover, (17) simplifies to an LP constraint satisfaction problem. For example, after squaring both sides of the first inequality $\|\lambda - \lambda^{k_j+1}\| \leq \|\lambda - \lambda^{k_j}\|$ within (17), after using the binomial expansion, and cancelling $\|\lambda - \lambda^{k_j}\|^2$ from both sides, the inequality simplifies to $2 \cdot (\lambda - \lambda^{k_j}) \cdot g(\tilde{x}^{k_j}, \tilde{y}^{k_j}) \geq s^{k_j} \cdot \|g(\tilde{x}^{k_j}, \tilde{y}^{k_j})\|^2$, which is linear in terms of $\lambda$.

To speed up convergence, however, a hyper-parameter $\zeta < 1$ will be introduced to reduce step-sizes as follows:

$$
s^k = \zeta \cdot \gamma \cdot \overline{q}_j - L(\tilde{x}^k, \tilde{y}^k, \lambda^k) \|g(\tilde{x}^k, \tilde{y}^k)\|^2, \zeta < 1.
$$

(23)
Subsequently, after iteration $k_{j+1}$ the problem (17) is sequentially solved again by adding one inequality per multiplier-updating iteration until iteration $k_{j+1}+n_{j+1}−1$ is reached for some $n_{j+1}$ so that (17) is infeasible. Then, step-size is updated by using $\tau_{j+1}$ per (23) and is used to update multipliers until the next time it is updated to $\tau_{j+2}$ when the “multiplier-divergence-detection” problem is infeasible again, and the process repeats.

**On Improvement of Convergence.** To speed up the acceleration of the multiplier-divergence detection through the “multiplier-divergence-detection” problem, (17) is modified, albeit heuristically, in the following way:

$$
\left\{\begin{array}{l}
\|\lambda - \lambda^{k_{j+1}}\| \leq \sqrt{1 - 2 \cdot \nu \cdot s_{k_{j}} \cdot \|\lambda - \lambda^{k_{j}}\|}, \\
\|\lambda - \lambda^{k_{j+2}}\| \leq \sqrt{1 - 2 \cdot \nu \cdot s_{k_{j}+1} \cdot \|\lambda - \lambda^{k_{j+1}}\|}, \\
\vdots \\
\|\lambda - \lambda^{k_{j+n_{j}}}\| \leq \sqrt{1 - 2 \cdot \nu \cdot s_{k_{j}+n_{j}-1} \cdot \|\lambda - \lambda^{k_{j+n_{j}-1}}\|}.
\end{array}\right.
$$

(24)

Unlike the problem (17), the problem (24) no longer simplifies to an LP problem. Nevertheless, the system of inequalities delineate the convex region and can still be handled by commercial software.

### 2.2 Numerical Testing

In this subsection, a series of examples is considered. In Example 2.2.1, large-scale instances of generalized assignment problems (GAPs) of types D and E with 20, 40 and 80 machines and 1600 jobs from the OR-library (https://www.or.amp.i.kyoto-u.ac.jp/members/yagiura/gap/) are considered to demonstrate efficiency, scalability, robustness, and competitiveness of the method with respect to the best results available thus far in the literature. In Example 2.2.2, a stochastic version of a job-shop scheduling problem instance with 127 jobs and 19 machines based on [16] is tested. In Example 2.2.3, two instances of pharmaceutical scheduling with 30 and 60 product orders, 17 processing units, and 6 stages based on [4] is tested.

#### 2.2.1 Example 1: Generalized Assignment Problems.

**Robustness of the Method.** The robustness with respect to initial step-sizes ($s^0$) is demonstrated in Table 1 using the GAP type D instance with 20 machines and 1600 jobs. Multipliers are initialized by using LP dual solutions. The following hyper-parameters are used: within the step-sizing formula (23), parameters $\gamma$ and $\zeta$ are chosen to be $\frac{1}{I}$ (the inverse of the number of machines) and $\frac{1}{5}$, respectively; within (24), parameter $\nu$ is chosen to be 2. As demonstrated in Table 1, the method’s performance is appreciably stable for the given range of initial step-sizes used.

**Table 1** Robustness Results for Instance d201600 with Respect to Initial Step-sizes.

| Initial Step-size ($s^0$) | Feasible Cost | Gap (%) | “Auxiliary” Time (sec) | Total Time (sec) |
|--------------------------|---------------|---------|------------------------|-----------------|
| 0.0025                   | **97825**     | 0.0037% | 123.71                 | 2427.71         |
| 0.005                    | **97825**     | 0.0037% | 6.84                   | 1226.17         |
| 0.01                     | 97826         | 0.0048% | 6.96                   | 2143.58         |
| 0.02                     | **97825**     | 0.0037% | 17.10                  | 1195.36         |
| 0.04                     | 97826         | 0.0048% | 19.21                  | 1941.55         |
SLBLR is robust with respect to initial multipliers (Table 2). For this purpose, the multipliers are initialized randomly by using the uniform distribution $U[90, 110]$. For the testing, the initial step-size $s^0 = 0.02$ was used. As evidenced from Table 2, the method’s performance is stable, exhibiting only a slight degradation of solution accuracy and increase of the CPU time as compared to the case with multipliers initialized by using LP dual solutions.

### Table 2: Robustness Results for Instance d201600 with Respect to Initial Multipliers.

| Case Number | Feasible Cost | Total Subproblem Solving Time (sec) | Feasible Solution Search Time (sec) | “Auxiliary” Time (sec) | Total Time (sec) |
|-------------|---------------|-------------------------------------|-------------------------------------|------------------------|-----------------|
| 1           | 97825         | 1098.74                             | 375.96                              | 22.13                  | 1496.84         |
| 2           | 97826         | 1009.42                             | 777.16                              | 173.48                 | 1960.07         |
| 3           | 97826         | 2223.99                             | 221.70                              | 4.54                   | 2450.24         |
| 4           | 97826         | 2333.55                             | 402.41                              | 4.08                   | 2740.04         |
| 5           | 97826         | 1002.77                             | 119.91                              | 160.73                 | 1283.42         |

### Scalability and Competitiveness of SLBLR.

To test scalability, six instances d201600, d401600, d801600, e201600, e401600 and e801600 are considered. SLBLR is compared with Depth-First Lagrangian Branch-and-Bound Method (DFLBnB) [18], Column Generation [19], and Very Large Scale Neighborhood Search (VLNS) [20], which to the best of the authors’ knowledge are the best methods for at least one of the above instances. For completeness, comparison against Surrogate Absolute-Value Lagrangian Relaxation (SAVLR) [17], which is an improved version of Surrogate Lagrangian Relaxation (SLR) [33], is also performed. The latter SLR method [33] has been previously demonstrated to be advantageous against other non-smooth optimization methods as explained in Section 4. As demonstrated in Table 3, the advantage of SLBLR is the ability to obtain optimal results across a wider range of GAP instances as compared to other methods. Even though the comparison in terms of the CPU time is not entirely fair, feasible-cost-wise, SLBLR decisively beats previous methods. For the d201600 instance, the results obtained by SLBLR and the Column Generation method [19] are comparable. For instance d401600, SLBLR obtains a better feasible solution and for instance d801600, the advantage over the existing methods is even more pronounced.

To the best of the authors’ knowledge, no other reported method obtained optimal results for instances d401600 and d801600. SLBLR outperforms SAVLR [17] as well, thereby demonstrating that the fast convergence offered by the novel “level-based” stepsizing, with other things being equal, translates into better results as compared to those obtained by SAVLR, which employs the “contraction mapping” stepsizing [17]. Lastly, the methods developed in [18–20] specifically target GAPs, whereas the SLBLR method developed in this paper has broader applicability.

#### 2.2.2 Example 2: Stochastic Job-Shop Scheduling with the Consideration of Scrap and Rework

Job shops are prevalent production environments designed for low-volume high-variety manufacturing. Within a job shop, a scheduling problem is to assign a set of jobs to a set of eligible machines by deciding what job needs to be processed on which machine group and at what time. Each job requires a specific sequence of
Table 3  Comparison Against the Best Results Currently Available.

| Instance | New Method | Posta [18] | Sadykov [19] | Haddadi [20] | Bragin [17] |
|----------|------------|------------|--------------|--------------|------------|
|          | (SLBLR)    | (DFLBnB)   | (Column Generation) | (VLSN)       | (SAVLR)    |
| d201600  | 97825 (1195) | -†         | 97825 (1026)    | 97836 (5364) | 97828 (1371) |
| d401600  | 97105* (836) | -†         | 97106 (919)     | 97125 (5364) | 97111 (1183) |
| d801600  | 97034* (3670)| -†         | 97037 (10860)   | 97075 (5364) | 97039 (1350) |
| e201600  | 180645** (85) |           |              | 180645 (749) |            |
| e401600  | 178293** (2478) | -         | 178293 (749)   |              |
| e801600  | 176820** (1762) |           | 176821 (749)   |              |

*The optimality is certified by the LP optimal values, which are 97105 and 97034 for instances d401600 and d801600, respectively.

**The optimality is certified through the lower bound results of, i.e., Posta et al. [18, p. 160].

†Not solved to optimality within 24 hours and not reported within the original paper of [18].

These instances were not considered within [19] and [17].
operations, and the processing time for each operation. As noted by [21, 22], even a single-machine scheduling is an NP-hard problem. To avoid late shipments to the greatest extent possible, tardiness is frequently minimized. This objective, however, requires the coordination of multiple jobs and multiple machines. Limited machine capacity brings a layer of difficulty since multiple “individual-job” subproblem are considered together competing for limited resources (machines). Another difficulty arises because of uncertainties, including processing times [23–28] and scrap [29–31]. Re-manufacturing of one part may affect and disrupt the overall schedule within the entire job shop, thereby leading to unexpectedly high delays in production.

![Figure 2](image.png)

**Fig. 2** The results for Example 2.2.2 are illustrated. SLBLR performs more than two orders of magnitude faster than branch-and-cut implemented in CPLEX.

The deterministic version of the job-shop problem from the paper of [16] has been solved within a few seconds by [32] by using CPLEX. The stochastic version of the problem (full problem formulation is available at [31]) with the consideration of scrap and rework requires 476.83 seconds to obtain an optimal solution by using CPLEX - two orders of magnitude increase of CPU time compared to a deterministic case, yet, the CPU time is still within the tolerable limits.

In this paper, we modified data from [16] by modifying several jobs by increasing a number of operations (e.g., from 1 to 6) and decreasing capacities of a few machines; the data are in Tables S1 and S2. These changes led to a drastic explosion of complexity even though the number of jobs stays the same and the number of machines decreases; as demonstrated in Figure 2, the running time of CPLEX increases to several days (from 476.83 seconds). In contrast, after relaxing machine capacity constraints and decomposing the relaxed problem into individual job subproblems, coordinating the subproblem solutions by using SLBLR, and at convergence heuristically adjusting beginning/completion times of selected jobs by a few time periods
until there are no machine capacity violations to obtain the overall feasible solution, a solution of the same quality as that obtained by CPLEX, is obtained within roughly 1 hour of CPU time.

2.2.3 Example 3: Multi-Stage Pharmaceutical Scheduling

As a third case example, we will evaluate the proposed method on a pharmaceutical scheduling problem proposed by [4]. Pharmaceutical manufacturing plants have high utilization, and effective scheduling is key to be able to deliver product orders on time. A unique aspect of these production plans is their sequence-dependence. Setup times vary based on the sequencing of products on each unit (machine). Scheduling in this context is combinatorial in the number of product orders, units, and stages. The new method is operationalized by relaxing constraints [4, Eqs. (1)-(3), p. 646] that couple individual processing units, namely assignment and processing time constraints. The results with SLBLR and Branch-and-Cut are demonstrated in Figures 3 and 4 below.

Fig. 3 The results for Example 2.2.3 with 30 products orders are illustrated. The SLBLR method’s performance is comparable to that of CPLEX.

With a small number of product orders, 30, an optimal solution with the feasible cost of 54.97 was found by CPLEX within 1057.78 seconds. The optimality is verified by running CPLEX until the gap is 0%; it took 171993.27 seconds to verify the optimality. SLBLR takes a slightly longer time to obtain the same solution - 1647.35 seconds.

In contrast, with 60 product orders, CPLEX no longer obtains good solutions in a computationally efficient manner; a solution with a feasible cost of 56.35 is obtained after 200,000 seconds. Within SLBLR, a solution with a feasible cost of 55.69 is obtained within 1978.04 seconds. This is more than two orders of magnitude of improvement over CPLEX. The drastic differences when the number of the
product orders doubles are demonstrated in Figure 4 (log scale). Moreover, for both cases, the time it takes SLBLR to match or beat the results obtained by CPLEX for both problem sizes is 1647.35 and 1978.035 seconds, respectively, signifying that the SLBLR is scalable.

![CPU Time Improvement: > 505 times](image)

**Fig. 4** The results for Example 2.2.3 with 60 products orders are illustrated. When doubling the number of products, CPLEX's performance is drastically deteriorated, while performance of SLBLR is largely unaffected.

### 3 Discussion

This paper develops the novel MILP solution methodology based on the Lagrangian Relaxation method. Salient features of the novel SLBLR method, inherited from previous version of Lagrangian Relaxation, are: 1. reduction of the computational effort required to obtain Lagrangian-multiplier-updating directions and 2. alleviation of zigzagging of multipliers. The key novel feature of the method, which the authors believe gives SLBLR the decisive advantage, is the innovative exploitation of the underlying geometric-convergence potential inherent to the Polyak's step-sizing formula without the heuristic adjustment of hyper-parameters for the estimate of \( q(\lambda^*) \) - the associated the “level” values are determined purely through the simple “auxiliary” constraint satisfaction problem. Herein lies the major advantage of SLBLR: the decision-based principle behind computing the “level” values, whereas the choice of hyper-parameters \( R \) and \( \delta \) within the incremental subgradient method (e.g., [14] or [35] discussed in Section 4) as well as the choice of \( M \) and \( r \) within the Surrogate-Lagrangian-Relaxation-based methods [17, 33] (discussed in Section 4) is problem-dependent and may require significant effort to determine appropriate values. Specifically, to obtain level values, heuristic adjustments of the “level” values are
required [14], based on multiplier “oscillation detection” or “significant descent” (for minimization of non-smooth functions). However, these rules do not allow to detect whether multipliers “start diverging.” Moreover, oscillation of multipliers is a natural phenomenon when optimizing non-smooth functions as discussed above since multipliers may zigzag/oscillate across ridges of the function, so the multiplier “oscillation detection” may not necessarily warrant the reduction of level values. On the other hand, multiplier “oscillation” is detected by checking whether multipliers travelled a (heuristically) predetermined distance \( R \), hence, the divergence of multipliers can go undetected for a significant number of iterations, depending on the value of \( R \). To the best of the authors’ knowledge, the subgradient- and surrogate-subgradient-based methods using Polyak’s (or Polyak-like) step-sizes with the intention of achieving the geometric/linear convergence rate either require \( q(\lambda^*) \), which is unavailable, or require multipliers to travel infinite distance to guarantee convergence to the optimum \( \lambda^* \) [14]. While SLR avoids the need to estimate \( q(\lambda^*) \), the geometric/linear convergence is only possible outside of a neighborhood of \( \lambda^* \) [33, p. 187]. Through testing, it is discovered that SLBLR is robust with respect to the choice of initial stepsizes and multipliers, computationally efficient, competitive and general. Several problems from diverse disciplines are tested and superiority of SLBLR is demonstrated. While “separable” MILP problems are considered, no particular problem characteristics such as linearity or separability have been used to obtain “level” values, and thus SLBLR has the potential to solve a broad class of MIP problems.

4 Methods: Previous Methods for Non-smooth Optimization and for MILP

The 1960s: Minimization of “Unsmooth Functionals.” The study of optimization of the class non-smooth convex functions, with \( q(\lambda) \) being its member, though irrespective to Lagrangian Relaxation, originates in Polyak’s seminal work ([36]). Polyak noted that subgradient directions are not always descending [36, p. 593] for minimization problems (accordingly, subgradient directions are not always ascending for maximization problems such as (3)). This property of subgradient at point A is shown in Figure 1. Nevertheless, convergence to the optimal solution optimizing the non-smooth function was proven under the following (frequently dubbed as “non-summable”) stepsizing formula satisfying the following conditions:

\[ s^k > 0, \quad \lim_{k \to \infty} s^k = 0, \quad \sum_{k=0}^{\infty} s^k = \infty. \quad (25) \]

A specific example of a “non-summable” step-size is \( s^k = s_0^{k^*} \), where \( s_0 \) is some positive constant.

The 1990s: The Subgradient-Level Method. The subgradient-level method, first proposed by [34] and subsequently improved by [14], overcomes the unavailability of the knowledge about the optimal value needed to compute Polyak’s step-size (7) by adaptively adjusting a “level” estimate based on detection of “sufficient descent” of the function and “oscillation” of solutions.

In terms of the problem (3), the procedure of the method is explained as follows: the “level” estimate \( q^k_{lev} = q^k_{rec} + \delta_j \) is used in place of the optimal dual value \( q(\lambda^*) \), where \( q^k_{rec} \) is the best dual value (“record objective value”) obtained up to an iteration \( k \), and \( \delta_j \) is an adjustable parameter with \( j \) denoting the \( j^{th} \) update of \( q^k_{lev} \). The main premise behind is when \( \delta_j \) is “too large,” then multipliers will exhibit
oscillations while travelling a significant (predefined) distance $R$ without improving the “record” value. In this case, the parameter $\delta_j$ is updated as $\delta_{j+1} = \beta \cdot \delta_j$ with $\beta = 1/2$. On the other hand, if $\delta_j$ is such that the dual value is sufficiently increased: $q(\lambda^k) \geq q_{lev}^{k} + \tau \cdot \delta_j$, with $\tau = 1/2$, then the parameter $\delta_j$ is unchanged and the distance travelled by multipliers is reset to 0 to avoid premature reduction of $\delta_j$ by $\beta$ in future iterations.

The Early 2000s: Incremental Subgradient Methods \cite{15, 35}. The main idea of the incremental subgradient method, like that of the interleaved method discussed above, is to improve convergence by solving a subproblem $i$ before updating multipliers. After one subgradient component is updated, rather than updating all the multipliers “at once,” within the incremental subgradient methods, multipliers are updated “incrementally.” After the $i^{th}$ subgradient component is calculated, the multipliers are incrementally updated as

$$\psi_i^k = \psi_{i-1}^k + s^k \cdot \left( A_i x_i^k - \beta_i \right).$$  \hspace{1cm} (26)

Here $\beta_i$ are the vectors such that $\sum_{i=1}^I \beta_i = b$, for example, $\beta_i = \frac{b}{I}$. Only after all $i$ subproblems are solved, are the multipliers “fully” updated as

$$\lambda^{k+1} = \psi_I^k.$$  \hspace{1cm} (27)

Convergence results of the subgradient-level method (\cite{14}) have been extended for the subgradient methods and proved. Variations of the method were proposed with $\beta$ and $\tau$ belonging to an interval $[0, 1]$ rather than being equal to $\frac{1}{2}$. Moreover, to improve convergence, rather than using constant $R$, a sequence of $R_i$ was proposed such that $\sum_{i=1}^\infty R_i = \infty$.

The 2010s: The Surrogate Lagrangian Relaxation Method \cite{33}. Convergence of the method is based on the “contraction mapping” concept. Namely, within the method, distances between multipliers at consecutive iterations are required to decrease, i.e.,

$$\|\lambda^{k+1} - \lambda^k\| = \alpha_k \cdot \|\lambda^k - \lambda^{k-1}\|, \hspace{1cm} 0 \leq \alpha_k < 1.$$  \hspace{1cm} (28)

Based on (9), the step-sizing formula has been derived:

$$s^k = \alpha_k \cdot \frac{s^{k-1} \|g(\hat{x}^{k-1})\|}{\|g(\hat{x}^k)\|}.$$  \hspace{1cm} (29)

Moreover, a specific formula to set $\alpha_k$ has been developed to guarantee convergence:

$$\alpha_k = 1 - \frac{1}{M \cdot k^{1-r}}, \hspace{1cm} M \geq 1, \hspace{1cm} 0 \leq r < 1.$$  \hspace{1cm} (30)

Within \cite{33}, Figs. 3-5,7; pp. 195-199], the SLR method showed the advantage against the above-mentioned subgradient-level method \cite{14} and the incremental subgradient methods \cite{15, 35} for non-smooth optimization.

Surrogate Absolute-Value Lagrangian Relaxation \cite{17}. Aiming to simultaneously guarantee convergence while ensuring fast reduction of constraint violations and preserving the linearity of the original MILP problem, the Surrogate Absolute-Value Lagrangian Relaxation (SAVLR) method was developed. Within the method, the following dual problem is considered:

$$\max_{\lambda} \{q_\rho(\lambda) : \lambda \in \Omega \subset \mathbb{R}^m\},$$  \hspace{1cm} (31)

where

$$q_\rho(\lambda) = \min_{x,y} \left\{ \sum_{i=1}^I (c_i^T x_i + c_i^y y_i) + \lambda^T \cdot \left( \sum_{i=1}^I A_i^T x_i + \sum_{i=1}^I A_i^y y_i - b \right) + \right.$$
\[
\rho \cdot \left\| \sum_{i=1}^{I} A_i^x x_i + \sum_{i=1}^{I} A_i^y y_i - b \right\|_1, \{ x_i, y_i \} \in F_i, i = 1, \ldots, I \right\}, \tag{32}
\]

The above minimization involves \(l_1\)-absolute-value piece-wise linear penalties, which efficiently penalize constraint violations and are exactly linearizable thereby enabling the use of MILP solvers. Within [17, Figs. 1, 2, 5-8; pp. 537-539], the SAVLR method showed the advantage over the above SLR method to solve GAPs. For completeness of comparison with the newly-developed SLBLR method, SAVLR is also included in Table 3.

Acknowledgements The work by M.A.B was supported in part by the US NSF under award ECCS-1810108.

Author Contributions M.A.B. conceptualized the project, developed the novel methodology, and conducted experiments. E.L.T. developed the third case study and supported testing and interpretation of results. M.A.B. drafted the initial manuscript with revisions from E.L.T. Both authors provided feedback on the final version. All authors read and approved the final manuscript.

Competing Interests The authors declare no competing interests.

Data Availability. Data supporting the results of Example 2.2.1 are located at https://www-or.amp.i.kyoto-u.ac.jp/members/yagiura/gap/; for Example 2.2.2, data are located in Tables S1 and S2 as well as in subsection 5.3; for Example 2.2.3, data are taken from [4].

References

[1] Liu, Q., Li, X., & Gao, L. Mathematical modeling and a hybrid evolutionary algorithm for process planning. *Journal of Intelligent Manufacturing* **32**(2), 781-797 (2021). https://doi.org/10.1007/s10845-020-01703-w

[2] Stefansson, H., Sigmarsdottir, S., Jensson, P., & Shah, N. Discrete and continuous time representations and mathematical models for large production scheduling problems: A case study from the pharmaceutical industry. *European Journal of Operational Research* **215**(2), 383–392 (2011). https://doi.org/10.1016/j.ejor.2011.06.021

[3] Ge, C., & Yuan, Z. Production scheduling for the reconfigurable modular pharmaceutical manufacturing processes. *Computers & Chemical Engineering* **151**, 107346 (2021). https://doi.org/10.1016/j.compchemeng.2021.107346

[4] Kopanos, G. M., Méndez, C. A., & Puigjaner, L. MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. *European Journal of Operational Research* **207**(2), 644–655 (2010). https://doi.org/10.1016/j.ejor.2010.06.002

[5] Zhu, S. X., & Ursavas, E. Design and analysis of a satellite network with direct delivery in the pharmaceutical industry. In *Transportation*
[6] Khelif Hachicha, H., & Zeghal Mansour, F. (2018). Two-MILP models for scheduling elective surgeries within a private healthcare facility. *Health Care Management Science* **21**(3), 376–392 (2018). https://doi.org/10.1007/s10729-016-9390-2

[7] Kayvanfar, V., Akbari Jokar, M. R., Rafiee, M., Sheikh, S., & Iranzad, R. A new model for operating room scheduling with elective patient strategy. In *INFOR: Information Systems & Operational Research* **59**(2), 309–332 (2021). https://doi.org/10.1080/03155986.2021.1881359

[8] Ahani, N., Andersson, T., Martinello, A., Teytelboym, A., & Trapp, A. C. Placement optimization in refugee resettlement. *Operations Research* **69**(5), 1468–1486 (2021). https://doi.org/10.1287/opre.2020.2093

[9] Huang, P. S., Boyken, S. E., & Baker, D. The coming of age of de novo protein design. *Nature* **537**, 320–327 (2016). https://doi.org/10.1038/nature19946

[10] Yang, L., Chen, R., Goodison, S., & Sun, Y. An efficient and effective method to identify significantly perturbed subnetworks in cancer. *Nature Computational Science* **1**, 79–88 (2021). https://doi.org/10.1038/s43588-020-00009-4

[11] Schill, W.-P., Pahle, M. & Gambardella, C. Start-up costs of thermal power plants in markets with increasing shares of variable renewable generation. *Nature Energy* **2**(6), 1-6 (2017). https://doi.org/10.1038/nenergy.2017.50

[12] Polyak, B. T. Minimization of unsmooth functionals. *USSR Computational Mathematics & Mathematical Physics* **9**(3), 14-29 (1969). https://doi.org/10.1016/0041-5553(69)90061-5

[13] Zhao, X., Luh, P. B., & Wang, J. Surrogate gradient algorithm for lagrangian relaxation. *Journal of Optimization Theory & Applications* **100**(3), 699-712 (1999). https://doi.org/10.1023/A:1022646725208

[14] Goffin, J.-L., & Kiwiel, K. C. Convergence of a simple subgradient level method. *Mathematical Programming* **85**, 207-211 (1999). https://doi.org/10.1007/s101070050053

[15] Nedić, A., & Bertsekas, D. Convergence rate of incremental subgradient Algorithms. In: Uryasev, S., Pardalos, P.M.(eds) *Stochastic Optimization: Algorithms and Applications*. *Applied Optimization* **54**, Springer, Boston, MA (2001).
Surrogate “Level-Based” LR for MILP

[16] Hoitomt, D. J., Luh, P. B., & Pattipati, K. R. A Practical approach to job shop scheduling problems. *IEEE Transactions on Robotics & Automation*, 9(1), 1-13 (1993). https://doi.org/10.1109/70.210791

[17] Bragon, M. A., Luh, P. B., Yan, B., & Sun, X. A Scalable solution methodology for mixed-integer linear programming problems arising in automation. *IEEE Transactions on Automation Science & Engineering*, 16(2), 531-541 (2019). https://doi.org/10.1109/TASE.2018.2835298

[18] Posta, M., Ferland, J. A., & Philippe, M. An exact method with variable fixing for solving the generalized assignment problem. *Computational Optimization & Applications*, 52(3), 629-644 (2012). https://doi.org/10.1007/s10589-011-9432-0

[19] Sadykov, R., Vanderbeck, F., Pessoa, A., & Uchoa, E. Column generation based heuristic for the generalized assignment problem. XLVII Simpósio Brasileiro de Pesquisa Operacional, Porto de Galinhas, Brazil, 3624-3631 (2015).

[20] Haddadi, S. Variable-fixing then subgradient optimization guided very large scale neighborhood search for the generalized assignment problem. *4OR – A Quarterly Journal of Operations Research*, 17(3), 261-295 (2019). https://doi.org/10.1007/s10288-018-0389-z

[21] Seidmann, A., Panwalkar, S. S., & Smith, M. L. Optimal assignment of due-dates for a single processor scheduling problem. *International Journal Of Production Research*, 19(4), 393-399 (1981), https://doi.org/10.1080/00207548108956667

[22] Panwalkar, S. S., Smith, M. L., & Seidmann, A. Common due date assignment to minimize total penalty for the one machine scheduling problem. *Operations Research*, 30(2), 391-399 (1982). https://doi.org/10.1287/opre.30.2.391

[23] Golenko-Ginzburg, D., & Gonik, A. Optimal job-shop scheduling with random operations and cost objectives. *International Journal of Production Economics*, 76(2), 147–157, ISSN 0925-5273 (2002). http://dx.doi.org/10.1016/S0925-5273(01)00140-2

[24] Lei, D. Simplified multi-objective genetic algorithms for stochastic job shop scheduling. *Applied Soft Computing*, 11(8), 4991–4996, ISSN 1568-4946 (2011). http://dx.doi.org/10.1016/j.asoc.2011.06.001

[25] Zhang, R., Song, S., & Wu, C. A hybrid differential evolution algorithm for job shop scheduling problems with expected total tardiness criterion. *Applied Soft Computing*, 13(3), 1448–1458 (2013). https://doi.org/10.1016/j.asoc.2012.02.024
[26] Shen, J., & Zhu, Y. Chance-constrained model for uncertain job shop scheduling problem. *Soft Computing* **20**(6), 2383–2391 (2016). https://doi.org/10.1007/s00500-015-1647-z

[27] Jamili, A. Job shop scheduling with consideration of floating breaking times under uncertainty. *Engineering Applications of Artificial Intelligence* **78**, 28–36 (2019). https://doi.org/10.1016/j.engappai.2018.10.007

[28] Horng, S. C., & Lin, S. S. Apply ordinal optimization to optimize the job-shop scheduling under uncertain processing times. *Arabian Journal for Science and Engineering* **47**, 9659–9671 (2022). https://doi.org/10.1007/s13369-021-06317-9

[29] Wilson, J. P., Shen, Z., Awasthi, U., Bollas, G. M., & Gupta, S.. Multi-objective optimization for cost-efficient and resilient machining under tool wear. *Journal of Advanced Manufacturing & Processing*, e10140 (2022). https://doi.org/10.1002/amp2.10140

[30] Sun, Y., Tu, J., Bragin, M. A., & Zhang, L. A Simulation-based integrated virtual testbed for dynamic optimization in smart manufacturing systems. *Journal of Advanced Manufacturing & Processing*, e10141 (2022). https://doi.org/10.1002/amp2.10141

[31] Bragin, M. A., Wilhelm, M. E., & Stuber, M. D. Toward agile and robust supply chains: A lesson from stochastic job-shop scheduling. Preprint at https://arxiv.org/abs/2206.09326 (2022).

[32] Liu, A., Luh, P. B., Yan, B., & Bragin, M. A. A novel integer linear programming formulation for job-shop scheduling problems. *IEEE Robotics & Automation Letters* **6**(3), 5937-5944 (2021). https://doi.org/10.1109/LRA.2021.3086422

[33] Bragin, M. A., Luh, P. B., Yan, J. H., Yu, N. & Stern, G. A. Convergence of the surrogate lagrangian relaxation method, *Journal of Optimization Theory & Applications* **164**(1), 173-201 (2015). https://doi.org/10.1007/s10957-014-0561-3

[34] Brännlund, U. On relaxation methods for nonsmooth convex optimization. *Ph.D. thesis*, Department of Mathematics, Royal Institute of Technology, Stockholm (1993).

[35] Nedić, A., & Bertsekas, D. P. Incremental subgradient methods for non-differentiable optimization. *SIAM Journal on Optimization* **12**(1), 109-138 (2001). https://doi.org/10.1137/S1052623499362111

[36] Polyak, B. T. A general method of solving extremum problems. *Sov. Math. Doklady (Soviet Mathematical Reports)* **8**(3), 593–597 (1967).
Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SupplementaryMaterial.pdf