Stabilized Column Generation for the Temporal Knapsack Problem using Dual-Optimal Inequalities

Timo Gschwind[a], Stefan Irnich[a]

[a] *Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*

Abstract

We present two new methods to stabilize column-generation algorithms for the Temporal Knapsack Problem (TKP). Caprara et al. [Caprara A, Furini F, and Malaguti E (2013) Uncommon Dantzig-Wolfe Reformulation for the Temporal Knapsack Problem. INFORMS J. on Comp. 25(3):560–571] were the first to suggest the use of branch-and-price algorithms for Dantzig-Wolfe reformulations of the TKP. Herein, the respective pricing problems are smaller-sized TKP that can be solved with a general-purpose MIP solver or by dynamic programming. Our stabilization methods are tailored to the TKP as they use (deep) dual-optimal inequalities, that is, inequalities known to be fulfilled by all (at least some) optimal dual solutions to the linear relaxation.

Key words: Column generation, dual inequalities, stabilization

1. Introduction

The *temporal knapsack problem* (TKP) is a generalization of the binary knapsack problem. It is defined by a set of *items* I and a discrete *time horizon* T. Each item has an associated *profit* p_i, a *weight* w_i, and the item is *active* at times $T_i \subseteq T$. Often TKP instances are defined by specifying a time window for each item, however, this assumption is not needed in the following. Conversely, let I_t be the set of the *items active* at time $t \in T$, i.e., $I_t = \{i \in I : t \in T_i\}$. Moreover, a *capacity* C is given. The TKP asks for a profit-maximizing subset of items such that at any time the weight of the selected active items does not exceed the capacity. A straightforward integer programming formulation for the TKP uses binary variables x_i, one for each item $i \in I$, to indicate that i is selected. It reads:

$$\max \quad \sum_{i \in I} p_i x_i \tag{1a}$$

$$\text{s.t.} \quad \sum_{i \in I_t} w_i x_i \le C \qquad \forall\, t \in T \tag{1b}$$

$$x_i \in \{0,1\} \qquad \forall\, i \in I \tag{1c}$$

As in (Caprara *et al.*, 2013) we assume that all coefficients are non-negative integers and that the time horizon T is already reduced so that only non-dominated constraints remain in (1b).

The TKP has appeared in the literature under different names: The name temporal knapsack problem was coined by Bartlett *et al.* (2005), who report applications in resource allocation problems which arise when bidding for a sparse resource such as for CPU time, communication bandwidth, computer memory, or disk space. They present solution algorithms combining techniques from constraint programming, artificial intelligence, and operations research. Caprara *et al.* motivate the TKP as a subproblem in a railway

Email addresses: gschwind@uni-mainz.de (Timo Gschwind), irnich@uni-mainz.de (Stefan Irnich)

application (Caprara *et al.*, 2011), where a train that travels along the stations $T = \{1, 2, \ldots, m\}$ can carry several railcars simultaneously if their total weight does not exceed C. One has to select among transportation requests, where the ith request consists of railcars of weight w_i to be transported from stations o_i to station d_i, i.e, along $T_i = \{o_i, o_i + 1, \ldots, d_i - 1, d_i\} \subseteq T$, providing a revenue of p_i if accepted. For additional references about applications in resource allocation, bandwidth allocation, and unsplittable flow on a line, we refer to (Caprara *et al.*, 2013, p. 560).

The basic idea of Caprara *et al.* (2013) was to partition the time horizon T into smaller so-called *blocks* and herewith to partition the constraints (1b) for the Dantzig-Wolfe reformulation of model (1). Typically, these blocks are time intervals of identical length. For example, if $T = \{0, 1, 2, \ldots, |T| - 1\}$, then the partitioning suggested is $T = \biguplus_{q\in Q} T_q$ with disjoint blocks of a chosen *block size* S given by $T_q = \{qS, qS+1, \ldots, (q+1)S-1\}$, where the indices q are taken from $Q = \{0, 1, \ldots, \lceil |T|/S \rceil - 1\}$. Obviously, the last block is smaller than S whenever $|T|$ is no multiple of S. In order to simplify the notation, we define the *items active in the block* T_q as $I_q = \bigcup_{t\in T_q} I_t$. Moreover, for each $q \in Q$, let

$$\mathcal{P}_q = \operatorname{conv}\left\{ v \in \{0,1\}^{I_q} : \sum_{i\in I_t} w_i v_i \le C, t \in T_q \right\}$$

be the convex hull of the solutions of the smaller TKP for the qth block. This polyhedron is bounded so that every point in $\mathcal{P}_q$ can be represented as a convex combination of the extreme points $\mathcal{E}_q$ of $\mathcal{P}_q$. Now, the Dantzig-Wolfe reformulation of Caprara *et al.* (2013) results from the grouping of the constraints (1b) according to the block partitioning:

$$\max \quad \sum_{i\in I} p_i x_i \tag{2a}$$

$$\text{s.t.} \quad x_i - \sum_{v\in\mathcal{E}_q} v_i \lambda_v^q = 0 \qquad \forall\, q \in Q, i \in I_q \tag{2b}$$

$$\sum_{v\in\mathcal{E}_q} \lambda_v^q = 1 \qquad \forall\, q \in Q \tag{2c}$$

$$x_i \in \{0,1\} \qquad \forall\, i \in I \tag{2d}$$

$$\lambda_v^q \ge 0 \qquad \forall\, q \in Q, v \in \mathcal{E}_q \tag{2e}$$

We refer to this formulation as the *integer master program* (IMP). The original variables x_i remain in IMP. Their function is to model the objective (2a) and to ensure that selected solutions $v = (v_i)_{i\in I_q}$ from different blocks are compatible, which is ensured by the coupling constraints (2b). The convexity constraints (2c) guarantee that exactly one solution is selected for each block. The variable domains are stated in (2d) and (2e).

The contribution of the paper at hand is to derive two new types of column-generation stabilization methods and to empirically validate their efficacy using some large-scale benchmark sets for the TKP. Both methods proposed here are based on (deep) dual-optimal inequalities, i.e., sets of inequalities known to hold for at least one dual optimal solution of the linear relaxation of the column-generation master program. This stabilization technique was originally proposed and tested by Ben Amor *et al.* (2006) for cutting stock and bin packing problems.

The remainder of the paper is structured as follows: The next section focuses on solving the reformulation of the TKP via branch-and-price and discusses the techniques to stabilized the underlying column-generation process. Section 3 presents and discusses the computational results before final conclusions are drawn in Section 4.

2. Stabilized Column Generation

We start with a brief summary on how column generation works for the TKP as suggested by Caprara *et al.* (2013) before we explain the new stabilization methods. We assume that the reader is familiar with integer column-generation techniques as, e.g., discussed in (Lübbecke and Desrosiers, 2005) and (Desaulniers *et al.*, 2005).

2.1. Column Generation for TKP as suggested by Caprara et al. (2013)

The number of extreme points of the polyhedra $\mathcal{P}_q, q \in Q$ is generally huge so that the Dantzig-Wolfe formulation (2) cannot be solved directly. Instead, one starts with a proper subset of the variables and solves the linear relaxation known as the *restricted master program* (RMP). The task of the pricing problems, there is one for each block, is to generate one or several variables λ_v^q with (smallest) negative reduced cost, or to prove that no such variable exists. Let $\pi = (\pi_{qi})_{q\in Q, i\in I_q}$ be the dual variables to the coupling constraints (2b) and $\mu = (\mu_q)_{q\in Q}$ be the dual variables to the convexity constraints (2c). In the following, we will distinguish between these dual variables and their values, which are denoted by $\bar{\pi}$ and $\bar{\mu}$, respectively.

Given the dual values $(\bar{\pi}, \bar{\mu})$, the reduced cost of a variable λ_v^q is $rdc(\lambda_v^q) = -\sum_{i\in I_q} \bar{\pi}_{qi} v_i + \bar{\mu}_q$. The *pricing problem* for the qth block is therefore of the form

$$\max \sum_{i\in I_q} \bar{\pi}_{qi} v_i, \quad \text{s.t.} \quad v = (v_i)_{i\in I_q} \in \mathcal{P}_q.$$

This is indeed a smaller-sized TKP, in which only the subset I_q of items is considered and the original profits are replaced by block-specific profits $\bar{\pi}_{qi}$. In principle, this problem can either be solved with a general-purpose MIP solver, with a tailored combinatorial algorithm such as dynamic programming (DP), or recursively with branch-and-price. Caprara *et al.* (2013) studied the first two options and found that for larger block sizes S the MIP solver (CPLEX) performed best. However, dynamic programming becomes a faster alternative for smaller block sizes S. In the extreme case of $S = 1$, the pricing problem is a binary knapsack problem, for which DP-based methods are certainly the state of the art (Kellerer *et al.*, 2004). In any case, the generated variables are added to the RMP, which is then re-optimized, and the process is repeated until no more variables with negative reduced cost exist. The solution of the RMP constitutes an optimal solution to the linear relaxation of (2). If it is fractional, branching on the original x_i variables finally yields integer solutions.

The general tradeoff in the branch-and-price approach exploited by Caprara *et al.* is the following: the larger the block size, the smaller the integrality gap and the branch-and-bound search tree, but the larger and therewith harder and more time consuming the individual pricing problems. However, a larger block size means a smaller number of pricing problems, less column-generation iterations are needed, and overall less columns are priced out. In their experiments, Caprara *et al.* had chosen block sizes that are powers of two and found out that 32, 64, and 128 are reasonable block sizes for the tested benchmark set, see Section 3.

2.2. Stabilization by Dual-Optimal Inequalities

Column-generation approaches in practice may suffer from instability problems. The values of the dual variables may oscillate heavily before they finally converge to some optimal values. In many applications the master program solution consists of only a few dense columns so that a primal basis must include several other variables that are then at value zero meaning that the primal model can be highly degenerated. This often leads to rather small and non-improving LP pivots, known as the tailing-off effect (Gilmore and Gomory, 1961; Vanderbeck, 2005): Over many iterations, the generated columns produce nearly no improvement in the LP objective. In order to explicitly stabilize the dual values, algorithmic techniques like the box step method (Marsten *et al.*, 1975), bundle methods (Hiriart-Urruty and Lemaréchal, 1993), and tailored stabilization approaches have been proposed (du Merle *et al.*, 1999; Rousseau *et al.*, 2007; Lee and Park, 2011). Furthermore, some recently proposed techniques can help overcome or even benefit from primal degeneracy when solving huge LPs (Gauthier *et al.*, 2014; Desrosiers *et al.*, 2014).

Another stabilization technique was originally suggested for the cutting stock and bin packing problem by Valério de Carvalho (2005) and Ben Amor *et al.* (2006). Valid inequalities known to hold for optimal dual solutions can be added as additional variables to the corresponding primal column-generation formulation. The restriction of the dual space leads to less possible intermediate values for the dual multipliers so that in many cases an optimal dual solution is computed faster. The recent paper (Gschwind and Irnich, 2014) extends the results of Ben Amor *et al.* (2006) with respect to theory, applications, algorithms, and computational results. Its main focus is, however, on column-generation models with unit (or equal) costs for all columns. The paper at hand can therefore be seen as a companion paper proving that also formulations with non-unit costs or profits benefit from this kind of stabilization.

Following Ben Amor *et al.* (2006), any inequality which is fulfilled by every optimal dual solution is called *dual-optimal inequality* (DOI). Moreover, a set of inequalities is called *deep dual-optimal inequalities* (DDOIs) if at least one optimal dual solution satisfies all inequalities. Hence, DOIs are always DDOIs. Conversely, two sets of DDOIs together may not form DDOIs, since they may cut off the entire dual space. In (Gschwind and Irnich, 2014), we were able to fully characterize in which situations the original dual and primal formulations are equivalent to their extended versions, i.e., those to which dual inequalities or additional columns are added: Equivalence is given if and only if the dual inequalities are DDOIs. For details we refer to Proposition 1 in (Gschwind and Irnich, 2014).

We now turn our attention to the TKP case. As already mentioned by Caprara *et al.* (2013, p. 564), the 'solution times of the LP relaxation are greatly reduced by replacing the "=" by "≤" in constraints' (2b). The formulations are equivalent because 'the set of TKP solutions forms an independence system', i.e., given any feasible subset of items, every subset of it is also feasible. From a dual point of view, the solution space is restricted to non-negative values for π.

We can add that also the equalities (2c) can be replaced by "≤" inequalities. Hence, every optimal dual solution fulfills

$$\pi_{qi} \geq 0 \quad \forall\, q \in Q, i \in I_q \qquad \text{and} \qquad \mu_q \geq 0 \quad \forall\, q \in Q, \tag{3}$$

i.e., they are DOIs.

Deep Dual-Optimal Inequalities for Profits. Another property results when interpreting the dual variables π. The value $\bar{\pi}_{qi}$ is the marginal profit that results from adding the item i to the solution of the qth block. It is natural to suspect that the summation of these dual values over all blocks is identical to the real profit p_i of item i. Indeed, the equalities

$$\sum_{q \in Q: i \in I_q} \pi_{qi} = p_i \qquad \forall\, i \in I \tag{4}$$

form a system of DDOIs, i.e., there exists at least one optimal solution to the dual formulation of (2) that satisfies all these equalities. Due to the RHS equal to the profit we call them *DDOIs for profits* in the following. Note that we keep the term DDOIs even for the equalities because any equality can be re-written by two inequalities.

The DDOI property can be seen as follows: The linear relaxation of the IMP (2) replaces integrality by $0 \leq x_i \leq 1$. However, these bounds are already enforced by the coupling constraints (2b). Hence, the RMP can be reformulated with unrestricted continuous variables $x_i \in \mathbb{R}, i \in I$. These variables have a corresponding dual constraint of the form (4). The implementation of the DDOIs into the column-generation model is therefore trivial because on must simply alter the x_i variables into unconstrained variables. Since any optimal solution (x_i) to the linear relaxation of this extended formulation is also optimal to the linear relaxation of (2), the DDOI property follows from the equivalence stated in Proposition 1 in (Gschwind and Irnich, 2014).

The impact of the DDOIs on the dual space is significant in cases, where only relatively few items belong to more than one block. Clearly, each of the $|I|$ DDOIs reduces the dual space by one dimension, while the dimension of the π space is $\sum_{q \in Q} |I_q|$. If only relatively few items belong to more than one block, the second number is approximately $|I|$.

Caprara *et al.* (2013) noted that it would be possible to reformulate the IMP (2) without any x_i variables for $i \in I$. In this case, the column-generation variables λ_v^q must replace the objective (2a), e.g., using the term $\sum_{v \in \mathcal{E}_q} p_i v_i \lambda_v^q$ for any chosen $q \in Q$ (or any convex combination of these terms for all $q \in Q$). Moreover, the coupling constraints (2b) would have to be substituted by constraints that enforce compatible solutions between pairs of blocks. Caprara *et al.* (2013, p. 563) mentioned that the formulation with the x_i variables, the so-called *explicit master*, worked better (see also Poggi de Aragao and Uchoa, 2003). Our explanation is that keeping $x_i \geq 0$ (as done by the authors) is beneficial as it partially stabilizes the dual variables using *in*equalities $\sum_{q\in Q:i\in I_q} \pi_{qi} \geq p_i$ for all $i \in I$ instead of equalities (4).

Deep Dual-Optimal Inequalities for Item Pairs. A second type of DDOIs results from comparing the dual values of two items $h, k \in I_q$ for the coupling constraints (2b) of the qth block. Recall that $\bar{\pi}_{qh}$ and $\bar{\pi}_{qk}$ are the marginal profits of including the items h and k in the qth block. One would suspect that $\bar{\pi}_{qh} \geq \bar{\pi}_{qk}$ holds for optimal dual solutions $(\bar{\pi}, \bar{\mu})$ if h is 'harder' to include than item k. Thereby, hardness should refer to both the weights, i.e., $w_h \geq w_k$, and the points in time when the items are active relative to the block T_q, i.e, $T_h \cap T_q \supseteq T_k \cap T_q$.

For $q \in Q$, we define *pairs of items for a replacement* as

$$R^q = \{(h,k) \in I_q \times I_q : w_h \geq w_k, T_h \cap T_q \supseteq T_k \cap T_q, \text{ and } w_h + w_k > C\}.$$

The last condition is hard to motivate, but it will turn out to be essential for the proof that the inequalities

$$\pi_{qh} \geq \pi_{qk} \qquad \forall\, q \in Q, (h,k) \in R^q \tag{5}$$

are DDOIs. The formal proof can be found in Section Appendix A of the Appendix. In the following, the DDOIs (5) are denoted as *DDOIs for item pairs.*

The resulting primal column-generation formulation will include additional variables y^q_{hk} for each $q \in Q, (h,k) \in R^q$. Its linear relaxation is:

$$\max \quad \sum_{i\in I} p_i x_i \tag{6a}$$

$$\text{s.t.} \quad x_i - \sum_{v\in\mathcal{E}_q} v_i \lambda^q_v + \sum_{q\in Q}\left(\sum_{k:(i,k)\in R^q} y^q_{ik} - \sum_{h:(h,i)\in R^q} y^q_{hi}\right) = 0 \qquad \forall\, q \in Q, i \in I_q \tag{6b}$$

$$\sum_{v\in\mathcal{E}_q} \lambda^q_v = 1 \qquad \forall\, q \in Q \tag{6c}$$

$$0 \leq x_i \leq 1 \qquad \forall\, i \in I \tag{6d}$$

$$\lambda^q_v \geq 0 \qquad \forall\, q \in Q, v \in \mathcal{E}_q \tag{6e}$$

$$y^q_{hk} \geq 0 \qquad \forall\, q \in Q, (h,k) \in R^q \tag{6f}$$

Compared to the IMP (2), the difference is in the coupling constraints (6b), in which the new non-negative variables y^q_{hk} occur (nonnegativity is ensured by (6f)). These variables have a rather intuitive interpretation: Imagine a solution with only one y variable in a block positive, say $y^q_{hk} = 1$, and the block solution v given by $\lambda^q_v = 1$. It means that in the solution of the qth block, the item h is replaced by item k. Surely, constraint (6b) for $i = h$ requires $v_h = 1$ and $x_h = 0$. In addition, the constraint (6b) for $i = k$ then imposes $v_k = 0$ and $x_k = 1$. Hence, the block solution $v_h = 1, v_k = 0$ is flipped to $x_h = 0, x_k = 1$. This interpretation of the DDOI variables is very similar to the what happens in the cutting stock and bin packing problem, where a positive stabilization variable in the extended column-generation formulation is also a replacement of one item by another item in a chosen bin, see (Ben Amor *et al.*, 2006).

Even if both (4) and (5) are DDOIs as individual set of inequalities, their union does not provide DDOIs. If the bounds (6d) would be relaxed (which happens when unconstrained primal variables x_i are introduced due to (4)), simultaneous replacements of the form $y_{hk} = y_{h'k} = 1$ with $h \neq h'$ would become possible. Conversely, simultaneous replacements $y_{hk} = y_{hk'} = 1$ with $k \neq k'$ would allow that two items are created out of the single item h. Both is not valid. Therefore, DDOIs (4) and (5) cannot be used together.

3. Computational Results

In this section, we summarize the computational experiments that we have conducted to compare the performance of the different column-generation algorithms to the TKP. We compare the otherwise identical branch-and-price algorithms that use different formulations as linear relaxations:

- `Base`: The linear relaxation of formulation (2), in which the DOIs (3) are implemented as "$\leq$" inequalities in (2b) and (2c).
- `DDOIs Profits`: The linear relaxation `Base` supplemented by the DDOIs for profits (3).
- `DDOIs Pairs`: The linear relaxation (6) using DDOIs for item pairs, in which the DOIs (3) are implemented as in `Base` in (6b) and (6c).

Moreover, we compare with the branch-and-price algorithm of Caprara *et al.* (2013) whenever the information is available from their paper:

- `Caprara et al.`: The algorithm uses the same linear relaxation as `Base` except that the convexity constraints (2c) are kept as equalities. (That is at least we read from the article.)

All algorithms were implemented in C++ using CPLEX 12.5 with default settings to solve the MIP subproblems. The experiments were performed on a standard PC with an Intel(R) Core(TM) i7-2600 at 3.4 GHz with 16.0 GB main memory using a single thread only.

We base our experiments on the benchmark set introduced by Caprara *et al.* that consists of two subclasses `I` and `U` each comprising 10 groups with 10 instances per group. For details on the generation of the instances and on their characteristics we refer to (Caprara *et al.*, 2013). Note that by construction no DDOIs for item pairs exist for the `U` instances because the capacity C is chosen large so that no item pair $h, k \in I$ fulfills $w_h + w_k > C$. Results for the `U` instances are, therefore, only presented for the algorithms `Base` and `DDOIs Profits`. Moreover, following the findings of Caprara *et al.* (2013), we restrict our analysis to block sizes between 32 and 128, which provide the best results for the considered instances. The time limit was set to one hour.

Linear Relaxation Results. Table 1 summarizes our results for the linear relaxations. The columns report the following information:

lp The number of instances for which the LP bound was found within the time limit

abs [s] The average solution time in seconds

rel The geometric mean of the instance-wise solution times relative to algorithm `Base`

iter The geometric mean of the instance-wise number of column-generation iterations relative to algorithm `Base` (only those instances that are solved by all considered algorithms are included)

Note first that the `U` instances are significantly harder to solve than the `I` instances. Indeed, neither algorithm was able to reach the LP bound for most instances within the time limit of one hour. For a better comparability of the different algorithms developed in our paper we added linear relaxation results of the `U` instances with an extended time limit of four hours.

A comparison between the two base implementations `Caprara et al.` and `Base` is difficult. Table 1 shows that `Caprara et al.` produces within 1 hour significantly more LP results. Possible explanations are that our code is less efficient or that they used a slightly faster computer, a different numerical tolerance, or included some acceleration techniques that are not present in our implementation.

Table 1 reveals that for all tested block sizes S both DDOIs for profits and DDOIs for item pairs stabilize the column-generation process and reduce the number of iterations needed to reach the LP bound. In the case of DDOIs for profits this also leads to a significant reduction of the computation times, especially for the difficult `U` instances. Moreover, algorithm `DDOIs Profits` provides LP bounds for many more instances than algorithm `Base`. In contrast, the stabilization effect of the DDOIs for item pairs does not help to reduce the computation times. Indeed, for block sizes of 64 and 128 algorithm `DDOIs Pairs` is on average slightly slower than algorithm `Base`. A more detailed analysis of the computation times is depicted in Figure 1 showing the linear relaxation solution times of the different algorithms relative to algorithm `Base`.

	$S = 32$				$S = 64$				$S = 128$			
		time				time				time		
	lp	*abs* [s]	*rel*	*iter*	*lp*	*abs* [s]	*rel*	*iter*	*lp*	*abs* [s]	*rel*	*iter*
`I` *instances*												
`Caprara et al.`	100	56.3	?	?	100	102.4	?	?	99	267.7	?	?
`Base`	100	90.3	1.00	1.00	100	110.6	1.00	1.00	100	209.0	1.00	1.00
`DDOIs Profits`	100	75.1	0.79	0.74	100	91.6	0.78	0.76	100	134.6	0.59	0.70
`DDOIs Pairs`	100	86.1	0.95	0.82	100	146.4	1.17	0.96	100	310.0	1.29	0.98
`U` *instances* (time limit 1 hour)												
`Caprara et al.`	43	>2340.8	?	?	47	>2177.5	?	?	45	>2186.5	?	?
`Base`	29	2700.1	1.00	1.00	38	2474.0	1.00	1.00	38	2450.0	1.00	1.00
`DDOIs Profits`	60	1916.3	0.34	0.33	52	2053.0	0.38	0.68	48	2129.2	0.35	0.26
`U` *instances* (time limit 4 hours)												
`Base`	30	10273.8	1.00	1.00	38	9170.0	1.00	1.00	54	8148.5	1.00	1.00
`DDOIs Profits`	84	4488.4	0.19	0.28	71	6152.5	0.28	0.77	61	6794.9	0.31	0.35

Table 1: Linear relaxation results

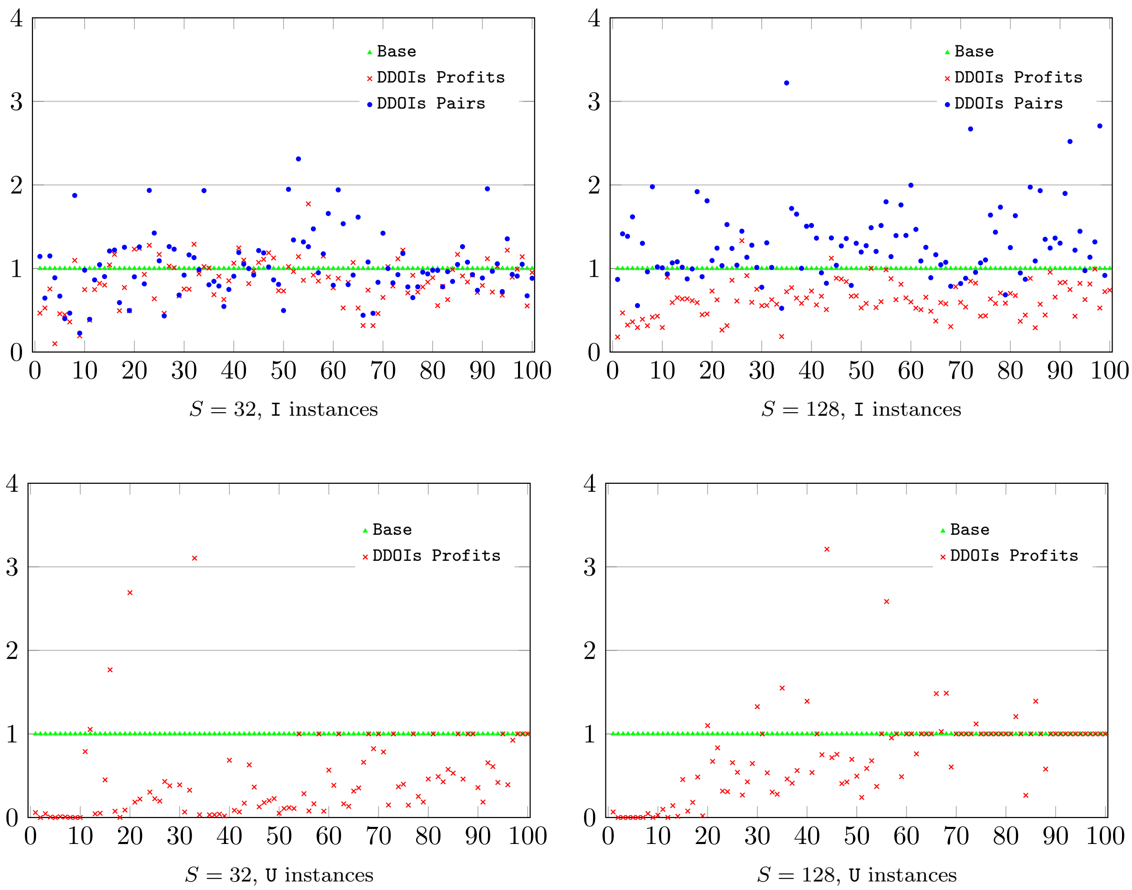

Figure 1: Computation times for linear relaxation relative to algorithm `Base`

	$S = 64$				$S = 96$				$S = 128$			
		time				time				time		
	opt	*abs* [s]	*rel*	*gap*	*opt*	*abs* [s]	*rel*	*gap*	*opt*	*abs* [s]	*rel*	*gap*
`I` instances												
`Base`	91	880.2	1.00	0.00	91	802.6	1.00	0.00	97	590.7	1	0.00
`DDOIs Profits`	96	537.4	0.60	0.00	97	484.2	0.63	0.00	97	390.0	0.60	0.00
`DDOIs Pairs`	86	1113.8	1.31	0.00	85	1025.7	1.24	0.00	92	813.1	1.33	0.00
`U` instances												
`Base`	27	2736.2	1.00	0.24	34	2613.7	1.00	0.15	34	2518.9	1	0.25
`DDOIs Profits`	40	2291.1	0.38	0.06	45	2154.9	0.36	0.11	45	2235.3	0.38	0.22

Table 2: Integer results

	`I` instances				*`U` instances*			
	opt	↑ *opt*	↑ *ub*	*gap*	*opt*	↑ *opt*	↑ *ub*	*gap*
`Caprara et al.`	96	0	0	0.00	42	1	4	0.14
`G & I`	100	4	4	0.00	51	10	54	0.03

Table 3: Summary comparison between Caprara *et al.* (2013) and our approach

Integer Results. Our integer solution results are summarized in Table 2. The additional columns report the number of instances that were solved to proven optimality within the time limit of one hour (*opt*) and the average remaining percentage gap with respect to the best known solution (*gap*). Note that in preliminary tests, we observed that the block size $S = 32$ is not competitive with block sizes of 64 and 128 and therefore chose not to include it in our integer solution analysis. Instead, we consider the additional block size $S = 96$ between 64 and 128.

From Table 2 it can be seen that the findings for the linear relaxation are transferable to integer solution results: Algorithm `DDOIs Profits` is clearly superior to algorithm `Base` with respect to computation times, number of solved instances, and remaining gap. The stabilization with DDOIs for items pairs, on the other hand, is not beneficial for the performance of the overall algorithm.

In Table 3, we summarize the comparison of our results with the results of Caprara *et al.* (2013). A more detailed, instance-wise comparison can be found in Tables B.4 and B.5 in Section Appendix B of the Appendix. Thereby, `Caprara et al.` refers to the strongest algorithm of Caprara *et al.* (2013) for a given instance. `G & I` has the analog meaning. We report the overall number of solved instances (*opt*), the number of instances that are solved to optimality only by the respective algorithm (↑ *opt*), the number of instances for which a stronger upper bound is provided by the algorithm (↑ *ub*), and the remaining average percentage integrality gap (*gap*).

Table 3 reveals that we are able to solve all `I` instances including the four instances that were previously unsolved. Regarding the `U` instances, Caprara *et al.* proved optimality for 42 instances, while we solve 51 instances. Thereby, we solve ten previously unsolved instances failing only on one of the instances that Caprara *et al.* solved. We found stronger upper bounds for 54 instances, while their upper bounds are stronger only for four instances. Moreover, our remaining gap of 0.03% is significantly smaller than the 0.14% of Caprara *et al.*.

4. Conclusions

In this paper we presented two types of column-generation stabilization methods for the temporal knapsack problem (TKP). Both methods are based on deep dual-optimal inequalities (DDOIs), namely, DDOIs for profits and DDOIs for item pairs. The integration of these dual inequalities as additional primal columns in the column-generation formulation reduces the oscillation of the dual values and herewith the number of column-generation iterations. While DDOIs for item pairs are not effective for the overall branch-and-price, the DDOIs for profits are: On average, measured by the geometric mean of runtime ratios, the new formulation with DDOIs for profits reduces the computation time of the branch-and-price by approximately 40% for the easier `I` instances of Caprara *et al.* and by approximately 60% for the harder `U` instances. In summary, using DDOIs for profits is very simple to implement, but effective, since the stabilized branch-and-price solves additional TKP instances to optimality, improves many upper bounds, and reduces the remaining gap for almost all open instances.

References

Bartlett, M., Frisch, A., Hamadi, Y., Miguel, I., Tarim, S., and Unsworth, C. (2005). The temporal knapsack problem and its solution. In R. Barták and M. Milano, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3524 of *Lecture Notes in Computer Science*, pages 34–48. Springer Berlin Heidelberg.

Ben Amor, H., Desrosiers, J., and Valério de Carvalho, J. M. (2006). Dual-optimal inequalities for stabilized column generation. *Operations Research*, **54**(3), 454–463.

Caprara, A., Malaguti, E., and Toth, P. (2011). A freight service design problem for a railway corridor. *Transportation Science*, **45**(2), 147–162.

Caprara, A., Furini, F., and Malaguti, E. (2013). Uncommon Dantzig-Wolfe reformulation for the temporal knapsack problem. *INFORMS Journal on Computing*, **25**(3), 560–571.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York, NY.

Desrosiers, J., Gauthier, J. B., and Lübbecke, M. E. (2014). Row-reduced column generation for degenerate master problems. *European Journal of Operational Research*, **236**(2), 453–460.

du Merle, O., Villeneuve, D., Desrosiers, J., and Hansen, P. (1999). Stabilized column generation. *Discrete Mathematics*, **194**, 229–237.

Gauthier, J. B., Desrosiers, J., and Lübbecke, M. E. (2014). Tools for primal degenerate linear programs. *EURO Journal on Transportation and Logistics*. (In press.).

Gilmore, P. and Gomory, R. (1961). A linear programming approach to the cutting-stock problem. *Operations Research*, **9**, 849–859.

Gschwind, T. and Irnich, S. (2014). Dual inequalities for stabilized column generation revisited. Technical Report LM-2014-03, Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University Mainz, Mainz, Germany.

Hiriart-Urruty, J.-B. and Lemaréchal, C. (1993). *Convex Analysis and Minimization Algorithms, Part 2: Advanced Theory and Bundle Methods*, volume 306 of *Grundlehren der Mathematischen Wissenschaften*. Springer, Berlin, Germany.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer, Berlin.

Lee, C. and Park, S. (2011). Chebyshev center based column generation. *Discrete Applied Mathematics*, **159**(18), 2251–2265.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Marsten, R., Hogan, W., and Blankenship, J. (1975). The boxstep method for large-scale optimization. *Operations Research*, **23**, 389–405.

Poggi de Aragao, M. and Uchoa, E. (2003). Integer program reformulation for robust branch-and-cut-and-price algorithms. In *Proc. Conf. Math. Program in Rio: A Conference in Honour of Nelson Maculan*, pages 56–61, Rio de Janeiro, Brazil.

Rousseau, L.-M., Gendreau, M., and Feillet, D. (2007). Interior point stabilization for column generation. *Operations Research Letters*, **35**(5), 660–668.

Valério de Carvalho, J. M. (2005). Using extra dual cuts to accelerate column generation. *INFORMS Journal on Computing*, **17**(2), 175–182.

Vanderbeck, F. (2005). Implementing mixed integer column generation. In Desaulniers *et al.* (2005), chapter 12, pages 331–358.

Appendix

A. Proof

In this section, we give a formal proof that the inequalities (5) are deep dual-optimal inequalities (DDOIs). Before providing the actual proof, we introduce some additional notation, which is useful for argumentation (cf. Ben Amor *et al.*, 2006; Gschwind and Irnich, 2014).

A pair of primal and dual LPs is

$$(P)\qquad \begin{aligned} z_P = &\min c^\top \lambda \\ \text{s.t.}\quad & A\lambda = b \\ & \lambda \geq 0 \end{aligned} \qquad\qquad (D)\qquad \begin{aligned} z_D = &\max b^\top \pi \\ \text{s.t.}\quad & A^\top \pi \leq c \end{aligned}$$

with a coefficient matrix $A = (a_{ij}) \in \mathbb{R}^{I\times J}$, cost coefficients $c = (c_i) \in \mathbb{R}^J$, and RHS $b = (b_i) \in \mathbb{R}^I$ with row indices $i \in I$ and column indices $j \in J$.

In the following, we assume that the primal formulation P is the extensive formulation to which a column-generation algorithm is applied. The idea of Ben Amor *et al.* (2006) was to add additional constraints $E^\top \pi \leq e$ to the dual D. Such additional constraints in the dual D correspond with additional variables in the primal P, denoted by y in the following. The extended primal and dual models are:

$$(\tilde{P})\qquad \begin{aligned} z_{\tilde{P}} = &\min c^\top \lambda + e^\top y \\ \text{s.t.}\quad & A\lambda + Ey = b \\ & \lambda \geq 0, y \geq 0 \end{aligned} \qquad\qquad (\tilde{D})\qquad \begin{aligned} z_{\tilde{D}} = &\max b^\top \pi \\ \text{s.t.}\quad & A^\top \pi \leq c \\ & E^\top \pi \leq e. \end{aligned}$$

The set of additional *dual inequalities* (DIs) $E^\top \pi \leq e$ cuts part of the dual solution space. Thus, $\tilde{D}$ is a restriction of D, whereas the corresponding primal model $\tilde{P}$ is a relaxation of P. We denote by D^* the set of optimal solutions to the model D, i.e., the dual-optimal space.

The following equivalence is stated and proven in (Gschwind and Irnich, 2014, Proposition 1). Equivalent are:

(i) $E^\top \pi \leq e$ are DDOIs.
(ii) There exists a $\pi^* \in D^*$ which is feasible also for $\tilde{D}$.
(iii) $z_D = z_{\tilde{D}}$.
(iv) $z_P = z_{\tilde{P}}$.
(v) For every feasible primal solution $(\tilde{\lambda}, y)$ to $\tilde{P}$ there exists a primal feasible solution λ to P with $c^\top \lambda \leq c^\top \tilde{\lambda} + e^\top y$.
(vi) There exists a primal optimal solution $(\tilde{\lambda}^*, y^*)$ to $\tilde{P}$ with $Ey^* = 0$. Also, $\tilde{\lambda}^*$ is an optimal solution to P.
(vii) Every optimal dual solution π^* to $\tilde{D}$ is optimal for D.

Proof that (5) are DDOIs. We will utilize the implication (v) $\Rightarrow$ (i). Thus, let P be the linear relaxation of (2), and let $\tilde{P}$ be the extended primal model (6). Note that in the formulations (2) and (6) the primal variables are x_i for $i \in I$ and λ_v^q for $q \in Q, v \in \mathcal{E}_q$. In the following, just one type of primal variables λ subsumes the x and λ variables in P and $\tilde{P}$. Moreover, in order to make the model (6) fit with formulation $\tilde{P}$, equality constraints can be established by introducing slack and surplus variables into (6d). Also these additional slack and surplus variables are subsumed in the λ variables of $\tilde{P}$. Furthermore, the maximization objectives in (2) and (6) can be transformed into a minimization objectives by setting $c_i = -p_i$ for the variables x_i.

The additional dual inequalities $E^\top \pi \leq e$ are the dual inequalities (5), here restated as

$$\pi_{qk} - \pi_{qh} \leq 0 \qquad \forall\, q \in Q, (h,k) \in R^q.$$

We see $e = \mathbf{0}$ in our case. Moreover, every additional primal column, i.e., the coefficients of the variable y_{hk}^q for $q \in Q, (h,k) \in R^q$, has cost zero and exactly two non-zero entries: the entry $+1$ in the row indexed with q and h, and the entry -1 in the row indexed with q and k. From now on, we will call the pair (h,k) a *valid replacement* for the qth block (for a more general definition of valid replacements we refer to (Gschwind and Irnich, 2014)).

Assume now that a feasible primal solution $((\tilde{x}, \tilde{\lambda}), y)$ to the extended model (6) is given. We have to show that there exists a primal feasible solution (x, λ) to the linear relaxation of formulation (2) with $c^\top x \leq c^\top \tilde{x}$, i.e., property (v). The latter condition is equivalent to $p^\top x \geq p^\top \tilde{x}$ and we will show equality.

If $y = 0$ there is nothing to do because $(\tilde{x}, \tilde{\lambda})$ is feasible for (2) with identical profit and hence also optimal for (2). Otherwise, there is at least one positive component of y, say $y^q_{hk} > 0$ corresponding to a valid replacement (h, k) for the qth block. A replacement cycle $(i_1, i_2, \ldots, i_p, i_1)$ for the qth block (with the additional definition $i_{p+1} := i_1$) is a cyclic sequence of different items such that (i_s, i_{s+1}) is a valid replacement and $y^q_{i_s i_{s+1}} > 0$ for all $s = 1, 2, \ldots, p$. A basic solution to (6) cannot contain any replacement cycles, since the corresponding columns are linear dependent. Consequently, all valid replacements (h, k) form a *directed acyclic graph* (DAG).

We first show that it is possible to construct, with the help of $((\tilde{x}, \tilde{\lambda}), y)$, another feasible solution $((\tilde{x}', \tilde{\lambda}'), y')$ to the extended model (6) with identical profit and $\mathbf{1}^\top y' < \mathbf{1}^\top y$. The latter inequality means that we can reduce the infeasibility w.r.t. formulation (2). Now, choose a longest path $P = (h = i_1, i_2, \ldots, i_p = i)$ in the DAG. Let $\varepsilon := \min\{y^q_{i_1 i_2}, y^q_{i_2 i_3}, \ldots, y^q_{i_{p-1} i_p}\} > 0$. The maximality of P implies $(Ey)_h \geq \varepsilon > 0$. The coupling constraint (6b) for h and q now imposes that some variables $\tilde{\lambda}^q_v$ with $v_h = 1$ are positive. Indeed, $\sum_{v \in : v_h = 1} \tilde{\lambda}^q_v \geq \varepsilon > 0$ holds. By definition of the valid replacements, i.e., the definition of set R^q, any $v \in \mathcal{P}_q$ with $v_h = 1$ must fulfill $v_{i_2} = v_{i_3} = \ldots = v_{i_p} = 0$. This results from $w_{i_1} \geq w_{i_2} \geq \cdots \geq w_{i_p}$ and $w_{i_{p-1}} + w_{i_p} > C$. In particular, for each such v it follows $v - u_h + u_i \in \mathcal{P}_q$, where u_h is the hth and u_i the ith unit vector. Since $y^q_{i_1 i_2} + y^q_{i_2 i_3} + \cdots + y^q_{i_{p-1} i_p} = u_h - u_i$, we can construct the new solution $(\tilde{x}', \tilde{\lambda}')$ as follows:

Initialize $((\tilde{x}', \tilde{\lambda}'), y')$ as $((\tilde{x}, \tilde{\lambda}), y)$ and $\varepsilon' := \varepsilon$. Iterate over the positive variables $\tilde{\lambda}'^q_v > 0$ with $v_h = 1$. Let $\tilde{\lambda}'^q_v$ be the variable under consideration, and let $\delta := \min\{\varepsilon', \tilde{\lambda}'^q_v\}$. Redefine

$$\begin{aligned} \tilde{\lambda}'^q_v &:= \tilde{\lambda}'^q_v - \delta, \\ \tilde{\lambda}'^q_{v-u_h+u_i} &:= \tilde{\lambda}'^q_{v-u_h+u_i} + \delta, \\ \varepsilon' &:= \varepsilon' - \delta. \end{aligned}$$

Terminate, when $\varepsilon' = 0$. (Termination with $\varepsilon' = 0$ is ensured because of $\sum_{v \in : v_h = 1} \tilde{\lambda}^q_v \geq \varepsilon$.) Finally, redefine

$$\begin{aligned} y'^q_{i_1 i_2} &:= y'^q_{i_1 i_2} - \varepsilon, \\ y'^q_{i_2 i_3} &:= y'^q_{i_2 i_3} - \varepsilon, \\ &\vdots \\ y'^q_{i_{p-1} i_p} &:= y'^q_{i_{p-1} i_p} - \varepsilon. \end{aligned}$$

Note that this new solution has identical cost and $\mathbf{1}^\top y' < \mathbf{1}^\top y$ holds.

By construction $((\tilde{x}', \tilde{\lambda}'), y')$ is feasible for the extended model (6). This type of update must be repeated (choosing a longest path in the DAG of a block) as long as $y' > 0$. The iterative updates finally result in $y' = 0$ because the replacement procedure eliminates at least one arc from the DAG for every chosen longest path P. This concludes the proof.

□

B. Extended Computational Results

Tables B.4 and B.5 present an instance-wise comparison of our results with the results reported in Caprara *et al.* (2013). `Caprara et al.` refers to the strongest algorithm of Caprara *et al.* (2013) for a given instance. `G & I` has the analog meaning. We report the name of the instance (*inst*), the best known solution (*lb*), the upper bounds (*ub*), and if the instance is solved to proven optimality (*opt*).

		Caprara et al.		G & I				Caprara et al.		G & I	
inst	*lb*	*ub*	*opt*	*ub*	*opt*	*inst*	*lb*	*ub*	*opt*	*ub*	*opt*
I1	62524	62524.00	*	62524.00	*	I51	71998	71998.00	*	71998.00	*
I2	65046	65046.00	*	65046.00	*	I52	81898	81898.00	*	81898.00	*
I3	67558	67558.00	*	67558.00	*	I53	97056	97056.00	*	97056.00	*
I4	70316	70316.00	*	70316.00	*	I54	107491	107491.00	*	107491.00	*
I5	76634	76634.00	*	76634.00	*	I55	120505	120505.00	*	120505.00	*
I6	77204	77204.00	*	77204.00	*	I56	129053	129053.00	*	129053.00	*
I7	81690	81690.00	*	81690.00	*	I57	142486	142486.00	*	142486.00	*
I8	84581	84581.00	*	84581.00	*	I58	151489	151489.00	*	151489.00	*
I9	87297	87297.00	*	87297.00	*	I59	165076	165097.00		165076.00	*
I10	88889	88889.00	*	88889.00	*	I60	182813	182813.00	*	182813.00	*
I11	88574	88574.00	*	88574.00	*	I61	22044	22044.00	*	22044.00	*
I12	96366	96366.00	*	96366.00	*	I62	26115	26115.00	*	26115.00	*
I13	97987	97987.00	*	97987.00	*	I63	29110	29110.00	*	29110.00	*
I14	103747	103747.00	*	103747.00	*	I64	32692	32692.00	*	32692.00	*
I15	103498	103498.00	*	103498.00	*	I65	37016	37016.00	*	37016.00	*
I16	108686	108686.00	*	108686.00	*	I66	39593	39593.00	*	39593.00	*
I17	112017	112017.00	*	112017.00	*	I67	44735	44735.00	*	44735.00	*
I18	116631	116631.00	*	116631.00	*	I68	48182	48182.00	*	48182.00	*
I19	125346	125346.00	*	125346.00	*	I69	50559	50559.00	*	50559.00	*
I20	128454	128454.00	*	128454.00	*	I70	54842	54842.00	*	54842.00	*
I21	87259	87259.00	*	87259.00	*	I71	40982	40982.00	*	40982.00	*
I22	89548	89548.00	*	89548.00	*	I72	47914	47914.00	*	47914.00	*
I23	96418	96418.00	*	96418.00	*	I73	52447	52447.00	*	52447.00	*
I24	98019	98019.00	*	98019.00	*	I74	59790	59790.00	*	59790.00	*
I25	104227	104227.00	*	104227.00	*	I75	66179	66179.00	*	66179.00	*
I26	107704	107704.00	*	107704.00	*	I76	75070	75070.00	*	75070.00	*
I27	109805	109805.00	*	109805.00	*	I77	81982	81982.00	*	81982.00	*
I28	116248	116248.00	*	116248.00	*	I78	85314	85314.00	*	85314.00	*
I29	119729	119729.00	*	119729.00	*	I79	95037	95037.00	*	95037.00	*
I30	123463	123463.00	*	123463.00	*	I80	100031	100031.00	*	100031.00	*
I31	102424	102424.00	*	102424.00	*	I81	71426	71426.00	*	71426.00	*
I32	103159	103159.00	*	103159.00	*	I82	82942	82942.00	*	82942.00	*
I33	111884	111884.00	*	111884.00	*	I83	96115	96115.00	*	96115.00	*
I34	117903	117903.00	*	117903.00	*	I84	110102	110102.00	*	110102.00	*
I35	120668	120668.00	*	120668.00	*	I85	119233	119233.00	*	119233.00	*
I36	123739	123739.00	*	123739.00	*	I86	128178	128178.00	*	128178.00	*
I37	130308	130308.00	*	130308.00	*	I87	142056	142056.00	*	142056.00	*
I38	133092	133092.00	*	133092.00	*	I88	154745	154770.00		154745.00	*
I39	138613	138613.00	*	138613.00	*	I89	167916	167916.00	*	167916.00	*
I40	144612	144612.00	*	144612.00	*	I90	176884	176916.00		176884.00	*
I41	30866	30866.00	*	30866.00	*	I91	42685	42685.00	*	42685.00	*
I42	35771	35771.00	*	35771.00	*	I92	46526	46526.00	*	46526.00	*
I43	40934	40934.00	*	40934.00	*	I93	54437	54437.00	*	54437.00	*
I44	46180	46180.00	*	46180.00	*	I94	60719	60719.00	*	60719.00	*
I45	50324	50324.00	*	50324.00	*	I95	68432	68432.00	*	68432.00	*
I46	55495	55495.00	*	55495.00	*	I96	72337	72346.00		72337.00	*
I47	59255	59255.00	*	59255.00	*	I97	80122	80122.00	*	80122.00	*
I48	65465	65465.00	*	65465.00	*	I98	88460	88460.00	*	88460.00	*
I49	69530	69530.00	*	69530.00	*	I99	92380	92380.00	*	92380.00	*
I50	75756	75756.00	*	75756.00	*	I100	100915	100915.00	*	100915.00	*

Table 4: Extended results for I instances

		Caprara et al.		G & I				Caprara et al.		G & I	
inst	*lb*	*ub*	*opt*	*ub*	*opt*	*inst*	*lb*	*ub*	*opt*	*ub*	*opt*
U1	49797	49797.00	*	49797.00	*	U51	34771	34774.00		34771.00	*
U2	49490	49490.00	*	49490.00	*	U52	33827	33838.75		33831.00	
U3	49020	49020.00	*	49020.00	*	U53	33197	33216.00		33197.00	*
U4	48972	48972.00	*	48972.00	*	U54	32942	32955.33		32942.00	*
U5	50149	50149.00	*	50149.00	*	U55	33318	33325.88		33319.00	
U6	49466	49466.00	*	49466.00	*	U56	33424	33465.54		33429.50	
U7	50666	50666.00	*	50666.00	*	U57	33438	33440.50		33438.00	*
U8	49859	49859.00	*	49859.00	*	U58	32059	32113.75		32066.50	
U9	50358	50358.00	*	50358.00	*	U59	33881	33887.50		33887.00	
U10	49961	49961.00	*	49961.00	*	U60	32973	33010.36		32973.50	
U11	46266	46266.00	*	46266.00	*	U61	31464	31496.89		31475.30	
U12	45628	45628.00	*	45628.00	*	U62	30330	30335.20		30337.80	
U13	45531	45531.00	*	45531.00	*	U63	30704	30723.65		30720.80	
U14	45218	45218.00	*	45218.00	*	U64	31315	31353.36		31336.10	
U15	45978	45978.00	*	45978.00	*	U65	31177	31186.00		31182.70	
U16	45795	45795.00	*	45795.00	*	U66	31059	31067.50		31069.50	
U17	46471	46471.00	*	46471.00	*	U67	31761	31834.68		31765.00	
U18	45877	45877.00	*	45877.00	*	U68	30445	30459.71		30446.00	
U19	46356	46356.00	*	46356.00	*	U69	32038	32066.45		32046.00	
U20	46217	46217.00	*	46217.00	*	U70	31650	31705.00		31654.00	
U21	41946	41946.00	*	41946.00	*	U71	30320	30487.62		30350.20	
U22	41346	41346.00	*	41346.00	*	U72	30338	30495.40		30369.20	
U23	40694	40694.00	*	40694.00	*	U73	29963	30051.37		29986.10	
U24	40955	40955.00	*	40955.00	*	U74	29544	29626.54		29554.60	
U25	41235	41235.00	*	41235.00	*	U75	29835	29973.79		29845.60	
U26	41168	41168.00	*	41168.00	*	U76	30156	30319.61		30181.10	
U27	42054	42054.00	*	42054.00	*	U77	29790	29928.42		29822.00	
U28	41475	41475.00	*	41475.00	*	U78	29380	29479.91		29390.20	
U29	42277	42277.00	*	42277.00	*	U79	29666	29790.09		29670.50	
U30	41684	41684.00	*	41684.00	*	U80	29784	29935.58		29814.70	
U31	38685	38685.00	*	38685.00	*	U81	29451	29586.24		29488.90	
U32	38106	38106.00	*	38106.00	*	U82	28207	28307.75		28233.70	
U33	38067	38067.00	*	38067.00	*	U83	27855	27979.38		27873.50	
U34	37159	37159.00	*	37159.00	*	U84	29472	29560.59		29472.00	*
U35	37826	37826.00	*	37826.00	*	U85	28335	28453.58		28369.20	
U36	37488	37491.33		37488.00	*	U86	28564	28699.96		28614.70	
U37	38237	38236.75		38237.00	*	U87	28584	28718.29		28611.10	
U38	37372	37372.00	*	37372.00	*	U88	28445	28562.54		28475.60	
U39	38374	38384.00		38374.00	*	U89	29042	29144.33		29058.90	
U40	37965	37965.00	*	37965.00	*	U90	27916	28061.62		27957.20	
U41	35538	35538.00	*	35538.00	*	U91	27727	27872.82		27762.80	
U42	34934	34944.50		34941.40		U92	26630	26771.13		26666.90	
U43	35071	35071.00	*	35071.00	*	U93	27817	27916.20		27834.70	
U44	35596	35596.00	*	35596.00	*	U94	27316	27387.50		27347.40	
U45	35112	35113.00		35114.50		U95	27126	27251.84		27186.50	
U46	34529	34533.54		34530.50		U96	27485	27623.50		27511.20	
U47	35866	35868.50		35866.00	*	U97	27006	27111.65		27041.70	
U48	34564	34564.00	*	34564.00		U98	26904	27000.10		26936.80	
U49	35883	35883.00	*	35883.00	*	U99	28759	28877.14		28782.40	
U50	35311	35322.20		35311.00	*	U100	27297	27385.89		27344.30	

Table 5: Extended results for U instances