Indicator Constraints in Mixed-Integer Programming

Andrea Lodi

University of Bologna, Italy - andrea.lodi@unibo.it

Amaya Nogales-Gómez, Universidad de Sevilla, Spain

Pietro Belotti, FICO, UK

Matteo Fischetti, Michele Monaci, Domenico Salvagnin, University of Padova, Italy

Pierre Bonami, IBM, Spain

SCIP Workshop 2014 @ Berlin (Germany), October 2, 2014

Indicator (bigM's) constraints

- We consider the linear inequality

$$a^T x \leq a_0, \tag{1}$$

where $x \in \mathbb{R}^k$ and $(a, a_0) \in \mathbb{R}^{k+1}$ are constant.

Indicator (bigM's) constraints

- We consider the linear inequality

$$a^T x \leq a_0, \tag{1}$$

where $x \in \mathbb{R}^k$ and $(a, a_0) \in \mathbb{R}^{k+1}$ are constant.

- It is a well-known modeling trick in Mixed-Integer Linear Programming (MILP) to use a binary variable y multiplied by a sufficiently big (positive) constant M in order to deactivate constraint (1)

$$a^T x \leq a_0 + My. \tag{2}$$

Indicator (bigM's) constraints

- We consider the linear inequality

$$a^T x \leq a_0, \tag{1}$$

where $x \in \mathbb{R}^k$ and $(a, a_0) \in \mathbb{R}^{k+1}$ are constant.

- It is a well-known modeling trick in Mixed-Integer Linear Programming (MILP) to use a binary variable y multiplied by a sufficiently big (positive) constant M in order to deactivate constraint (1)

$$a^T x \leq a_0 + My. \tag{2}$$

- It is also well known the risk of such a modeling trick, namely
  - weak Linear Programming (LP) relaxations, and
  - numerical issues.

Complementarity Reformulation

- An alternative for logical implications and general deactivations is given by the *complementary* reformulation

$$(a^T x - a_0)\bar{y} \leq 0, \tag{3}$$

where $\bar{y} = 1 - y$. It has been used for decades in the Mixed-Integer Nonlinear Programming literature (MINLP).

Complementarity Reformulation

- An alternative for logical implications and general deactivations is given by the *complementary* reformulation

$$(a^T x - a_0)\bar{y} \leq 0, \tag{3}$$

where $\bar{y} = 1 - y$. It has been used for decades in the Mixed-Integer Nonlinear Programming literature (MINLP).

- The obvious drawback of the above reformulation is its *nonconvexity*.
- Thus, the complementary reformulation has been used so far if (and only if) the problem at hand was already nonconvex, as it is often the case, for example, in Chemical Engineering applications.

Our goal

- In this talk we argue against this common rule of always pursuing a linear reformulation for logical implications.

Our goal

- In this talk we argue against this common rule of always pursuing a linear reformulation for logical implications.

- We do that by exposing a class of Mixed-Integer Convex Quadratic Programming (MIQP) problems arising in *Supervised Classification* where the Global Optimization (GO) solver `Couenne` using reformulation (3) is out-of-the-box consistently faster than virtually any state-of-the-art commercial MIQP solver like `IBM-Cplex`, `Gurobi` and `Xpress`.

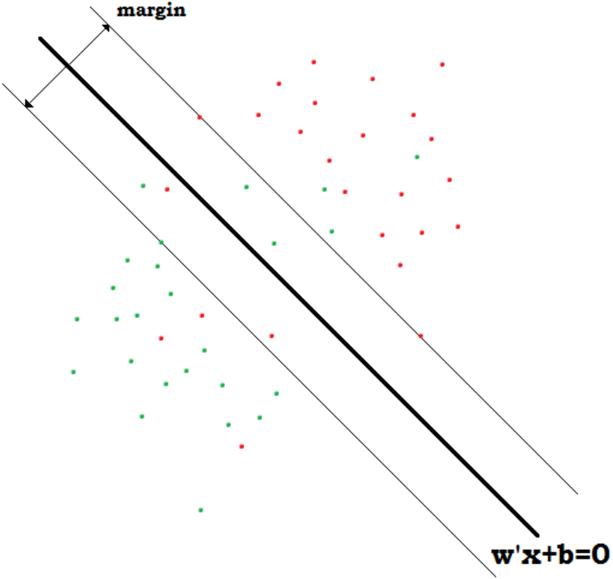
Our goal

- In this talk we argue against this common rule of always pursuing a linear reformulation for logical implications.

- We do that by exposing a class of Mixed-Integer Convex Quadratic Programming (MIQP) problems arising in *Supervised Classification* where the Global Optimization (GO) solver `Couenne` using reformulation (3) is out-of-the-box consistently faster than virtually any state-of-the-art commercial MIQP solver like `IBM-Cplex`, `Gurobi` and `Xpress`.

- This is quite counter-intuitive because, in general, convex MIQPs admit more efficient solution techniques both in theory and in practice, especially because they benefit of virtually all machinery of MILP solvers.

Support Vector Machine (SVM)



The input data

- Ω: the population.
- Population is partitioned into two classes, $\{-1, +1\}$.
- For each object in Ω, we have
  - $x = (x^1, \ldots, x^d) \in X \subset \mathbb{R}^d$: predictor variables.
  - $y \in \{-1, +1\}$: class membership.

- The goal is to find a hyperplane $\omega^\top x + b = 0$ that aims at separating, if possible, the two classes.
- Future objects will be classified as

$$\begin{aligned} y = +1 \quad & \text{if} \quad \omega^\top x + b > 0 \\ y = -1 \quad & \text{if} \quad \omega^\top x + b < 0 \end{aligned} \tag{4}$$

Soft-margin approach

$$\min \frac{\omega^\top \omega}{2} + \sum_{i=1}^{n} g(\xi_i)$$

subject to

$$\begin{array}{rcll} y_i(\omega^\top x_i + b) & \geq & 1 - \xi_i & i = 1, \ldots, n \\ \xi_i & \geq & 0 & i = 1, \ldots, n \\ \omega \in \mathbb{R}^d, b \in \mathbb{R} & & & \end{array}$$

where n is the size of the population and $g(\xi_i) = \frac{C}{n}\xi_i$ the most popular choice for the loss function.

Ramp Loss Model (Brooks, *OR*, 2011)

- Ramp Loss Function $g(\xi_i) = (\min\{\xi_i, 2\})^+$ yielding the Ψ-learning approach, with $(a)^+ = \max\{a, 0\}$.

Ramp Loss Model (Brooks, *OR*, 2011)

- Ramp Loss Function $g(\xi_i) = (\min\{\xi_i, 2\})^+$ yielding the Ψ-learning approach, with $(a)^+ = \max\{a, 0\}$.

$$\min \frac{\omega^\top \omega}{2} + \frac{C}{n}\left(\sum_{i=1}^{n} \xi_i + 2\sum_{i=1}^{n} z_i\right)$$

s.t. (RLM)

$$\begin{aligned} y_i(\omega^\top x_i + b) &\geq 1 - \xi_i - M z_i & \forall i = 1, \ldots, n \\ 0 \leq \xi_i &\leq 2 & \forall i = 1, \ldots, n \\ z &\in \{0,1\}^n & \\ \omega \in \mathbb{R}^d, b &\in \mathbb{R} & \end{aligned}$$

with $M > 0$ big enough constant.

Expectations (and Troubles)

- In principle, RLM is a tractable Mixed-Integer Convex Quadratic Problem that nowadays commercial (and even some noncommercial) solvers should be able to solve:
  - convex objective function,
  - linear constraints, and
  - binary variables,

  not much more difficult than a standard Mixed-Integer *Linear* Problem.

Expectations (and Troubles)

- In principle, RLM is a tractable Mixed-Integer Convex Quadratic Problem that nowadays commercial (and even some noncommercial) solvers should be able to solve:
  - convex objective function,
  - linear constraints, and
  - binary variables,

  not much more difficult than a standard Mixed-Integer *Linear* Problem.
- However, the bigM constraints in the above model destroy the chances of the solver to consistently succeed for $n > 50$.

Expectations (and Troubles)

- In principle, RLM is a tractable Mixed-Integer Convex Quadratic Problem that nowadays commercial (and even some noncommercial) solvers should be able to solve:
  - convex objective function,
  - linear constraints, and
  - binary variables,

  not much more difficult than a standard Mixed-Integer *Linear* Problem.
- However, the bigM constraints in the above model destroy the chances of the solver to consistently succeed for $n > 50$.
- We consider 23 instances from Brooks, `Type B`, $n = 100$, time limit of 3,600 CPU seconds.

Solving the MIQP by `IBM-Cplex`

- 23 instances from Brooks, Type B, $n = 100$, time limit of 3,600 CPU seconds.
- `IBM-Cplex` is able to find the optimal solution (%gap of the upper bound, ub) **BUT**
- it fails to improve the continuous relaxation by terminating after 1h with a large %gap for the lower bound, lb.

IBM-Cplex			
		% gap	
time (sec.)	nodes	ub	lb
3,438.49	16,142,440	–	–
tl	12,841,549	–	23.61
tl	20,070,294	–	37.82
tl	20,809,936	–	9.37
tl	17,105,372	–	26.17
tl	13,865,833	–	22.67
tl	14,619,065	–	21.40
tl	13,347,313	–	14.59
tl	12,257,994	–	22.22
tl	13,054,400	–	23.13
tl	14,805,943	–	12.37
tl	12,777,936	–	21.97
tl	14,075,300	–	23.32
tl	13,994,099	–	12.48
tl	10,671,225	–	23.08
tl	12,984,857	–	22.72
tl	12,564,000	–	14.11
tl	11,217,844	–	23.45
tl	12,854,704	–	22.72
tl	14,018,831	–	12.43
tl	11,727,308	–	23.55
tl	15,482,162	–	18.67
tl	12,258,164	–	14.88

Reformulating by Complementarity

$$\min \frac{\omega^\top \omega}{2} + \frac{C}{n}\left(\sum_{i=1}^{n} \xi_i + 2\sum_{i=1}^{n}(1-\bar{z}_i)\right)$$

$$(y_i(\omega^\top x_i + b) - 1 + \xi_i)\cdot\bar{z}_i \geq 0 \qquad \forall i = 1,\dots,n$$

$$0 \leq \xi_i \leq 2 \qquad \forall i = 1,\dots,n$$

$$\bar{z} \in \{0,1\}^n$$

$$\omega \in \mathbb{R}^d$$

$$b \in \mathbb{R},$$

where $\bar{z}_i = 1 - z_i$, and

- the resulting model is a Mixed-Integer Nonconvex Quadratically Constrained Problem (MIQCP) that `IBM-Cplex`, like all other commercial solvers initially developed for MILP, cannot solve (yet).

Solving the MIQCP by Couenne

- Despite the nonconvexity of the above MIQCP, there are several options to run the new model as it is and one of them is the open-source solver Couenne belonging to the Coin-OR arsenal.

Solving the MIQCP by Couenne

- Despite the nonconvexity of the above MIQCP, there are several options to run the new model as it is and one of them is the open-source solver Couenne belonging to the Coin-OR arsenal.

Couenne			
		% gap	
time (sec.)	nodes	ub	lb
163.61	17,131	–	–
1,475.68	181,200	–	–
tl	610,069	14.96	15.38
160.85	25,946	–	–
717.20	131,878	–	–
1,855.16	221,618	–	–
482.19	56,710	–	–
491.26	55,292	–	–
1,819.42	216,831	–	–
807.95	89,894	–	–
536.40	62,291	–	–
1,618.79	196,711	–	–
630.18	83,676	–	–
533.77	65,219	–	–
2,007.62	211,157	–	–
641.05	72,617	–	–
728.93	73,142	–	–
1,784.93	193,286	–	–
752.50	84,538	–	–
412.16	48,847	–	–
2,012.62	223,702	–	–
768.73	104,773	–	–
706.39	70,941	–	–

What does Couenne do?

- Although,
  - Convex MIQP should be much easier than nonconvex MIQCP, and
  - `IBM-Cplex` is by far more sophisticated than `Couenne`

  one can still argue that a comparison in performance between two different solution methods and computer codes is anyway hard to perform.

What does Couenne do?

- Although,
  - Convex MIQP should be much easier than nonconvex MIQCP, and
  - `IBM-Cplex` is by far more sophisticated than `Couenne`

  one can still argue that a comparison in performance between two different solution methods and computer codes is anyway hard to perform.

- However, the reported results are rather surprising, especially if one digs into the way in which `Couenne` solves the problem, namely considering three aspects:
  1. McCormick Linearization,
  2. Branching, and
  3. alternative L_1 norm.

McCormick Linearization

- The most crucial observation is that the complementarity constraints are internally reformulated by `Couenne` through the classical McCormick linearization
  1. $\vartheta_i = y_i(\omega^\top x_i + b) - 1 + \xi_i$, with $\vartheta_i^L \leq \vartheta_i \leq \vartheta_i^U$, and
  2. $u_i = \vartheta_i \bar{z}_i$

  for $i = 1, \ldots, n$.

McCormick Linearization

- The most crucial observation is that the complementarity constraints are internally reformulated by `Couenne` through the classical McCormick linearization
  1. $\vartheta_i = y_i(\omega^\top x_i + b) - 1 + \xi_i$, with $\vartheta_i^L \leq \vartheta_i \leq \vartheta_i^U$, and
  2. $u_i = \vartheta_i \bar{z}_i$

  for $i = 1, \ldots, n$. Then, the product corresponding to each new variable u_i is linearized as

$$
\begin{aligned}
u_i &\geq 0 && (5)\\
u_i &\geq \vartheta_i^L \, \bar{z}_i && (6)\\
u_i &\geq \vartheta_i + \vartheta_i^U \, \bar{z}_i - \vartheta_i^U && (7)\\
u_i &\leq \vartheta_i + \vartheta_i^L \, \bar{z}_i - \vartheta_i^L && (8)\\
u_i &\leq \vartheta_i^U \, \bar{z}_i && (9)
\end{aligned}
$$

  again for $i = 1, \ldots, n$, where (5) are precisely the complementarity constraints and ϑ_i^L and ϑ_i^U play the role of the bigM.

Branching

- It is well known that a major component of GO solvers is the iterative tightening of the convex (most of the time linear) relaxation of the nonconvex feasible region by branching on continuous variables.

Branching

- It is well known that a major component of GO solvers is the iterative tightening of the convex (most of the time linear) relaxation of the nonconvex feasible region by branching on continuous variables.

- However, the default version of `Couenne` does not take advantage of this possibility and branches (first) on binary variables $\bar{z}$'s.

- Thus, again it is surprising that such a branching strategy leads to an improvement over the sophisticated branching framework of `IBM-Cplex`.

Alternative L_1 norm

- A natural question is if the reported results are due to the somehow less sophisticated evolution of MILP solvers in their MIQP extensions with respect to the MILP one.

Alternative L_1 norm

- A natural question is if the reported results are due to the somehow less sophisticated evolution of MILP solvers in their MIQP extensions with respect to the MILP one.

- In order to answer this question we performed an experiment in which the quadratic part of the objective function has been replaced by its L_1 norm making the entire bigM model linear. Ultimately, the absolute value of ω is minimized.

Alternative L_1 norm

- A natural question is if the reported results are due to the somehow less sophisticated evolution of MILP solvers in their MIQP extensions with respect to the MILP one.
- In order to answer this question we performed an experiment in which the quadratic part of the objective function has been replaced by its L_1 norm making the entire bigM model linear. Ultimately, the absolute value of ω is minimized.
- Computationally, this has no effect and Couenne continues to consistently outperform MILP solvers on this very special (modified) class of problems.

Tightening ω's based on the objective function

- Bound reduction is a crucial tool in MINLP: it allows one to eliminate portions of the feasible set while guaranteeing that at least one optimal solution is retained.

Tightening ω's based on the objective function

- Bound reduction is a crucial tool in MINLP: it allows one to eliminate portions of the feasible set while guaranteeing that at least one optimal solution is retained.
- Among those reductions, we observed that `Couenne` does a very simple bound tightening (at the root node) based on the computation of an upper bound, i.e., a feasible solution, of value, say, U

$$\omega_i \in \left[-\sqrt{2U}, \sqrt{2U}\right] \qquad \forall i = 1, \ldots, d.$$

Tightening ω's based on the objective function

- Bound reduction is a crucial tool in MINLP: it allows one to eliminate portions of the feasible set while guaranteeing that at least one optimal solution is retained.
- Among those reductions, we observed that `Couenne` does a very simple bound tightening (at the root node) based on the computation of an upper bound, i.e., a feasible solution, of value, say, U

$$\omega_i \in \left[-\sqrt{2U}, \sqrt{2U}\right] \qquad \forall i = 1, \ldots, d.$$

- We did implement this simple bound tightening in `IBM-Cplex` and it is already very effective by triggering further propagation on binary variables (i.e., fixings)

Tightening ω's based on the objective function

- Bound reduction is a crucial tool in MINLP: it allows one to eliminate portions of the feasible set while guaranteeing that at least one optimal solution is retained.
- Among those reductions, we observed that `Couenne` does a very simple bound tightening (at the root node) based on the computation of an upper bound, i.e., a feasible solution, of value, say, U

$$\omega_i \in \left[-\sqrt{2U}, \sqrt{2U}\right] \qquad \forall i = 1, \ldots, d.$$

- We did implement this simple bound tightening in `IBM-Cplex` and it is already very effective by triggering further propagation on binary variables (i.e., fixings) but only if the initial bigM values are tight enough.

Tightening ω's based on the objective function

- Bound reduction is a crucial tool in MINLP: it allows one to eliminate portions of the feasible set while guaranteeing that at least one optimal solution is retained.
- Among those reductions, we observed that `Couenne` does a very simple bound tightening (at the root node) based on the computation of an upper bound, i.e., a feasible solution, of value, say, U

$$\omega_i \in \left[-\sqrt{2U}, \sqrt{2U}\right] \qquad \forall i = 1, \dots, d.$$

- We did implement this simple bound tightening in `IBM-Cplex` and it is already very effective by triggering further propagation on binary variables (i.e., fixings) but only if the initial bigM values are tight enough.
- In other words, when the bigM values are large it is very hard to solve the problem without changing them during search.

Much more sophisticated propagation

- It has to be noted that `Couenne` internal bigM values (namely ϑ_i^L and ϑ_i^U) are much more conservative (and safe) than those used in the SVM literature.

Much more sophisticated propagation

- It has to be noted that `Couenne` internal bigM values (namely ϑ_i^L and ϑ_i^U) are much more conservative (and safe) than those used in the SVM literature.
- Nevertheless, the sophisticated bound reduction loop implemented by GO solvers does the job.

Much more sophisticated propagation

- It has to be noted that `Couenne` internal bigM values (namely ϑ_i^L and ϑ_i^U) are much more conservative (and safe) than those used in the SVM literature.
- Nevertheless, the sophisticated bound reduction loop implemented by GO solvers does the job. Iteratively,
  - new feasible solutions propagate on the ω variables,
  - that leads to strengthen MC constraints (by changing the ϑ_i bounds),
  - that in turn propagates on binary variables.

Much more sophisticated propagation

- It has to be noted that Couenne internal bigM values (namely ϑ_i^L and ϑ_i^U) are much more conservative (and safe) than those used in the SVM literature.
- Nevertheless, the sophisticated bound reduction loop implemented by GO solvers does the job. Iteratively,
    - new feasible solutions propagate on the ω variables,
    - that leads to strengthen MC constraints (by changing the ϑ_i bounds),
    - that in turn propagates on binary variables.
    - Conversely, branching on the $\bar{z}_i$
    - either $(\bar{z}_i = 0)$ increases the lower bound,
      thus triggering additional ω tightening,
    - or $(\bar{z}_i = 1)$ tightens the bounds on ϑ_i, thus
      propagating again on ω.

Much more sophisticated propagation

- It has to be noted that Couenne internal bigM values (namely ϑ_i^L and ϑ_i^U) are much more conservative (and safe) than those used in the SVM literature.
- Nevertheless, the sophisticated bound reduction loop implemented by GO solvers does the job. Iteratively,
  - new feasible solutions propagate on the ω variables,
  - that leads to strengthen MC constraints (by changing the ϑ_i bounds),
  - that in turn propagates on binary variables.
  - Conversely, branching on the $\bar{z}_i$
  - either $(\bar{z}_i = 0)$ increases the lower bound,
    thus triggering additional ω tightening,
  - or $(\bar{z}_i = 1)$ tightens the bounds on ϑ_i, thus
    propagating again on ω.
- Switching off in Couenne any of these components leads to a dramatic degradation in the results.

Initial Bound tightening

- Bound reduction at the root node can make use of enumeration.

Initial Bound tightening

- Bound reduction at the root node can make use of enumeration.
- Iterative domain reduction
  - **while** (hope_to_improve) **do**
    for each $i \in [1, d]$ **do**
    $l_i = \min\{\omega_i : (w, \xi, \bar{z}_i) \in P, Z(w, \xi, \bar{z}_i) \le U\}$
    $u_i = \max\{\omega_i : (w, \xi, \bar{z}_i) \in P, Z(w, \xi, \bar{z}_i) \le U\}$
    where:
    - P is the set of feasible solutions (including bound constraints),
    - $Z(w, \xi, \bar{z}_i)$ is the cost of solution $(w, \xi, \bar{z}_i)$, and
    - U is the value of an upper bound.
- The MILPs need not to be solved to optimality: l_i and u_i are the bounds returned by the MIP within a node limit.
- Improve lower and upper bounds for ω_i variables $\rightarrow$ possibly fix some ω_i and/or z_i variables.

Iterated domain reduction

- Iterate domain reduction can be seen as a preprocessing tool.
- The better the initial solution U, the more effective the reduction.
- It can be time consuming, but dramatically improves the results:
  - Use `IBM-Cplex` for 100k nodes (+ 10 polish nodes) to compute the initial solution U.
  - Use `IBM-Cplex` for 100k nodes for each lower/upper bound tightening.
  - All instances with $d = 2$ solved to optimality:
    average CPU time $\simeq 30$ seconds
    average # of nodes: 412k
- Iterating the process allows `IBM-Cplex`'s internal preprocessing tools to propagate the current domain of the variables.

IBM-Cplex work in progress

		CPLEX 12.6.0	CPLEX w.i.p.	
	optimal value	time	time	time ratio
1	157,995.00	6799.8	253.8	0.04
2	179,368.00	10000.0	3467.2	0.35
3	220,674.00	10000.0	10000.0	1.00
4	5,225.99	10000.0	314.8	0.03
5	5,957.08	10000.0	10000.0	1.00
6	11,409,600.00	10000.0	7995.7	0.80
7	11,409,100.00	10000.0	521.9	0.05
8	10,737,700.00	10000.0	2300.0	0.23
9	5,705,360.00	10000.0	10000.0	1.00
10	5,704,800.00	10000.0	1015.0	0.10
11	5,369,020.00	10000.0	10000.0	1.00
12	2,853,240.00	10000.0	10000.0	1.00
13	2,852,680.00	10000.0	871.4	0.09
14	2,684,660.00	10000.0	2552.6	0.26
15	1,427,170.00	10000.0	10000.0	1.00
16	1,426,620.00	10000.0	2656.3	0.27
17	1,342,480.00	10000.0	10000.0	1.00
18	714,142.00	10000.0	3841.2	0.38
19	713,583.00	10000.0	942.3	0.09
20	671,396.00	10000.0	8653.7	0.87
21	357,626.00	10000.0	2608.0	0.26
22	357,067.00	10000.0	1094.2	0.11
23	335,852.00	10000.0	10000.0	1.00

Conclusions

- In a broad sense, we have used the SVM with the ramp loss to investigate the possibility of exploiting tools from (nonconvex) MINLP in MILP or (convex) MIQP, essentially, the reverse of the common path.

- More precisely, we have argued that sophisticated (nonconvex) MINLP tools might be very effective to face one of the most structural issues of MILP, which is dealing with the weak continuous relaxations associated with bigM constraints.

- We have shown that the crucial bound reduction can be obtained *a priori*, in this special case by solving MILPs on the weak bigM formulation.

- More generally, a full integration of such a bound reduction tool for indicator constraints can be obtained by local cuts.

Disclaimer

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion. Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

For example: Couenne branching on continuous

		Couenne default				Couenne continuous			
				% gap				% gap	
	optimal value	time (sec.)	nodes	ub	lb	time (sec.)	nodes	ub	lb
1	157,995.00	163.61	17,131	–	–	323.21	62,873	–	–
2	179,368.00	1,475.68	181,200	–	–	561.95	106,905	–	–
3	220,674.00	tl	610,069	14.96	15.38	490.29	134,758	–	–
4	5,225.99	160.85	25,946	–	–	238.26	65,152	–	–
5	5,957.08	717.20	131,878	–	–	535.70	142,368	–	–
6	11,409,600.00	1,855.16	221,618	–	–	773.62	149,880	–	–
7	11,409,100.00	482.19	56,710	–	–	985.26	195,438	–	–
8	10,737,700.00	491.26	55,292	–	–	535.78	103,806	–	–
9	5,705,360.00	1,819.42	216,831	–	–	726.18	143,234	–	–
10	5,704,800.00	807.95	89,894	–	–	1,031.75	172,794	–	–
11	5,369,020.00	536.40	62,291	–	–	546.78	109,142	–	–
12	2,853,240.00	1,618.79	196,711	–	–	663.69	127,318	–	–
13	2,852,680.00	630.18	83,676	–	–	790.88	160,010	–	–
14	2,684,660.00	533.77	65,219	–	–	510.01	99,802	–	–
15	1,427,170.00	2,007.62	211,157	–	–	717.69	117,670	–	–
16	1,426,620.00	641.05	72,617	–	–	932.44	161,835	–	–
17	1,342,480.00	728.93	73,142	–	–	512.60	83,890	–	–
18	714,142.00	1,784.93	193,286	–	–	720.15	119,761	–	–
19	713,583.00	752.50	84,538	–	–	983.23	168,276	–	–
20	671,396.00	412.16	48,847	–	–	449.68	86,351	–	–
21	357,626.00	2,012.62	223,702	–	–	661.69	110,343	–	–
22	357,067.00	768.73	104,773	–	–	706.15	156,464	–	–
23	335,852.00	706.39	70,941	–	–	493.32	79,719	–	–