HAL open science

Mixed Integer Programming formulations for the balanced Traveling Salesman Problem with a lexicographic objective

Luc Libralesso

To cite this version:

Luc Libralesso. Mixed Integer Programming formulations for the balanced Traveling Salesman Problem with a lexicographic objective. 2020. hal-02569456

HAL Id: hal-02569456

https://hal.science/hal-02569456v1

Preprint submitted on 11 May 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Mixed Integer Programming formulations for the balanced Traveling Salesman Problem with a lexicographic objective

Luc Libralesso[0000−0001−9908−4811]

Univ. Grenoble Alpes, CNRS, Grenoble INP, G-SCOP, 38000 Grenoble, France
luc.libralesso@grenoble-inp.fr

Abstract. This paper presents a Mixed Integer Program to solve the Balanced TSP. It exploits the underlying structure of the instances and is able to find optimal solutions for all the instances provided in the Metaheuristics Summer School competition. We study the efficiency of this new model on several variants of the Balanced TSP. The proposed method was ranked first in the MESS18 Metaheuristic competition among 9 submissions.
Instances and ranking: `http://195.201.24.233/mess2018/home.html`.
Source code: `https://gitlab.com/librallu/balancedtspcode`.

Keywords: Balanced TSP · MIP · Metaheuristics

1 Introduction, Problem description, and preliminary results

This paper presents a MIP formulation for a kind of optimization problem that involves the minimization of an absolute lexicographic value. It obtains optimality proofs for all instances of the Metaheuristic Summer School 2018 competition (`http://195.201.24.233/mess2018/home.html`) in a (relatively) short amount of time.

Consider a graph $G = (V, E)$, and a weight function $w : E \to \mathbb{Z}$. We want to find a Hamiltonian tour T that minimizes $|\sum_{e \in T} w_e|$.

We can formulate this problem as follows:

$$
\begin{aligned}
\min \quad & z \\
\text{subject to} \quad & \\
\sum_{j \in V,\, ij \in E} x_{ij} &= 2 && \forall i \in V && (1)\\
\sum_{i,j \in S,\, ij \in E} x_{ij} &\leq |S| - 1 && \forall S \subset V, S \neq \emptyset && (2)\\
z &\geq \sum_{e \in E} x_e w_e && && (3)\\
z &\geq -\sum_{e \in E} x_e w_e && && (4)\\
x_e &\in \{0,1\} && \forall e \in E \\
z &\in \mathbb{Z}
\end{aligned}
$$

Constraints (1) and (2) are tour constraints. Constraint (1) guarantees the degree of vertices to be 2 in the tour. Constraint (2) guarantees no subtour. Constraints (3) and (4) make z be the absolute value of the sum of edges in the tour.

If used as is, this formulation gives poor results (see Table 1). A significant improvement can be achieved by taking into account some properties of the instances presented in the competition.

1.1 A better MIP formulation

The MESS2018 competition[1] instances represent sparse graphs. Indeed, an instance of size n contains $4 \cdot n$ edges. It implies that the resulting MIP models contain a (relatively) small number of variables. Moreover, it turns out that each weight w can be written as $w = \overline{w} \cdot 10^5 + \underline{w}$, with $\overline{w} \in [0, 10]$ and $\underline{w} \in [0, 100]$. These weights are lexicographic since a solution that does not optimize the $\overline{w}$ part will be worse than a solution that optimizes $\overline{w}$ (unless there are many edges in the instance). Using this property, we can preprocess the input to decompose weights and derive a new model that introduces 3 objectives. One for the sum of $\overline{w}$, one for the sum of $\underline{w}$ and the last one for the sum of w.

[1] http://195.201.24.233/mess2018/home.html

$$
\begin{aligned}
\min \quad & z \\
\text{subject to} \quad & \\
\sum_{j \in V,\, ij \in E} x_{ij} &= 2 && \forall i \in V && (5)\\
\sum_{i,j \in S,\, ij \in E} x_{ij} &\leq |S| - 1 && \forall S \subset V, S \neq \emptyset && (6)\\
\overline{z} &= \sum_{e \in E} x_e \overline{w_e} && && (7)\\
\underline{z} &= \sum_{e \in E} x_e \underline{w_e} && && (8)\\
z &\geq \overline{z} \cdot 10^5 + \underline{z} && && (9)\\
z &\geq -(\overline{z} \cdot 10^5 + \underline{z}) && && (10)\\
x_e &\in \{0, 1\} && \forall e \in E \\
z, \overline{z}, \underline{z} &\in \mathbb{Z}
\end{aligned}
$$

Constraint (5) guarantees the degree of vertices to be 2 in the tour. Constraint (6) eliminates subtours. Constraints (7) and (8) define $\overline{z}$ (resp. $\underline{z}$) to be equal to the weighted sum of selected edges. Constraints (9) and (10) define z to be equal to the absolute value of the original weighted sum of edges.

In the remaining of this document, we call this model the *lifted MIP model*. We call the original model the *standard MIP model*.

1.2 Numerical results

Results were obtained on an Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz core with 8 GB RAM and Linux Mint 11. The MIP models were implemented using Gurobi 8.1 and python2.7. Table **??** reports optimal values, and computation times and gaps. In the dataset, instances are denoted by their size. For example, instance 0500 has 500 vertices. Column *optimal obj* shows the value obtained by the *lifted MIP model*. Column *time to opt* reports the time needed by the *lifted MIP model* to find the optimality proof. Finally columns *lb standard*, *ub standard*, *gap standard* ($gap = \frac{ub-lb}{ub}$) report bounds obtained by the *standard MIP model* within 30 seconds. We note that no more time is needed to observe a clear difference in performance between the two models.

As the results show, lifting the variable space in the second MIP has a dramatic effect on performance. Indeed, it allows the solver to branch in order to optimize first the $\overline{w}$ part of the problem, consequently improving the search speed. During the competition, I tried many matheuristic approaches (mostly local branching like). At some point, I noticed that the lifted approach was able to solve optimally the competition instances in a few seconds. It would be interesting to compare the lifted approach with other meta-heuristics (like local branching) on bigger instances (*i.e.* 5.000, 10.000 *etc.*).

instance	optimal objective	time to optimal (s)	lower bound standard	upper bound standard	gap standard (%)
0010	105	0	105	105	0.0
0015	271	0	271	271	0.0
0020	296	0	86	296	71.0
0025	375	0	87	386	77.5
0030	433	0	0	434	100.0
0040	458	0	0	604	100.0
0050	630	0	100	762	76.4
0060	821	0	0	948	100.0
0070	858	0	0	1.419	100.0
0080	807	0	5	1.324	100.0
0090	974	0	212	1.150	81.6
0100	996	0	266	2.228	88.1
0150	1.673	1	0	3.003	100.0
0200	2.029	0	0	3.838	100.0
0250	2.798	0	0	5.497	100.0
0300	3.695	2	119	7.240	98.4
0400	4.709	2	0	9.919	100.0
0500	5.747	5	0	9.983	100.0
0600	6.548	8	154	12.573	98.8
0700	8.097	8	0	16.169	100.0
0800	9.234	15	0	18.019	100.0
0900	9.271	12	0	19.481	100.0
1000	11.202	29	0	23.604	100.0
1500	16.339	51	0	36.304	100.0
2000	20.757	133	0	48.789	100.0
2500	1.333	510	0	35.075	100.0
3000	0	2.125	0	24.074	100.0

Table 1: Comparison of the standard and lifted formulations for the balanced TSP

In Appendix A, Figures 1, 2, 3, 4, 5 and 6 show the convergence curves of the *lifted MIP formulation*. We observe very different behaviours on the convergence curves. In the btsp0100 and btsp0250 instances, the upper bound converges quickly and the lower bound is found in the later stages of the resolution. In the btsp0500 instance, the dual bound converges quickly to some good value. However, there is still an important gap to optimality that is closed later during the resolution. In the btsp0700 instance, both best solution found and dual bound improve at the same time to some very small gap (less than 1%). Finally, we observe on btsp1000 and btsp1500 instances that finding a feasible solution takes time (sometimes half the resolution time) and the lower bound of good quality is found in the later stages of the resolution.

In this section, we investigated the lifting process. It allows a dramatic performance increase. This allows us to find optimal solutions for all instances of the benchmark. In the next section, we investigate this impact on different variants of the problem (namely assignment variant, subset variant, balanced spanning tree and regular TSP).

2 Impact on problem variants

We study different problem variants and evaluate the performance difference between the standard and the lifted model.

For each variant, we present a *standard MIP model*. We perform the lifting strategy as described on the balanced TSP. For the sake of simplicity, since the lifting process for each variant is totally similar to the balanced TSP, we do not present the lifted formulation alongside each standard formulation.

2.1 Balanced Assignment

A first related problem we may want to consider is the assignment problem. It consists of removing from the original problem the subtour elimination constraint. We note that it constitutes a relaxation of the original balanced TSP.

We define the balanced assignment problem as follows:

$$
\begin{aligned}
\min \quad & z \\
\text{subject to} \quad & \\
\sum_{j \in V,\, ij \in E} x_{ij} &= 2 && \forall i \in V && (11)\\
z &\geq \sum_{e \in E} x_e w_e && && (12)\\
z &\geq -\sum_{e \in E} x_e w_e && && (13)\\
x_e &\in \{0,1\} && \forall e \in E \\
z &\in \mathbb{Z}
\end{aligned}
$$

Constraint (11) forces each vertex to have two incident edges, constraint (12) and (13) force the balanced objective.

As for the balanced TSP, one can define the lifted version of the balanced assignment.

2.2 Balanced fixed-size subset

In this section, we describe a relaxation of the assignment problem. We relax the degree constraint on each vertex and add a constraint to fix the number of edges selected. The following model describes the subset problem:

$$
\begin{aligned}
\min \quad & z \\
\text{subject to} \quad & \\
\sum_{e \in E} x_e &= n && && (14)\\
z &\geq \sum_{e \in E} x_e w_e && && (15)\\
z &\geq -\sum_{e \in E} x_e w_e && && (16)\\
x_e &\in \{0,1\} && \forall e \in E \\
z &\in \mathbb{Z}
\end{aligned}
$$

Constraint (14) forces the number of edges selected, constraints (15), (16) define z as an objective. As for the previous models, we consider both the standard version and the lifted version.

2.3 TSP

Since the lifted formulation for the balanced-(TSP,assignment,subset) made a huge difference in the performance, we added some experiments for the classical

TSP. Without surprise, gurobi was able to handle even the big instances in a few seconds for both models. However, it seems that the lift formulation does not imply any gain in performance on the classical TSP. The following model presents the TSP formulation used:

$$
\begin{aligned}
\min \quad & z \\
\text{subject to} \quad & \\
\sum_{j \in V,\, ij \in E} x_{ij} &= 2 && \forall i \in V && (17)\\
\sum_{i,j \in S,\, ij \in E} x_{ij} &\leq |S| - 1 && \forall S \subset V, S \neq \emptyset && (18)\\
z &\geq \sum_{e \in E} x_e w_e && && (19)\\
x_e &\in \{0,1\} && \forall e \in E \\
z &\in \mathbb{Z}
\end{aligned}
$$

Constraints (17) and (18) are tour constraints. Constraint (19) sets the objective to the minimal tour value.

2.4 Balanced Spanning Tree

A related problem can be to find a balanced minimum spanning tree in the graph. We relax the degree constraint for each vertex. Also, we add a constraint that forces the number of selected edges to be $n-1$ and the graph to be connected. The following model describes the balanced spanning tree problem:

$$
\begin{aligned}
\min \quad & z \\
\text{subject to} \quad & \\
\sum_{e \in E} x_e &= n - 1 && && (20)\\
\sum_{i,j \in S,\, ij \in E} x_{ij} &\leq |S| - 1 && \forall S \subset V, S \neq \emptyset && (21)\\
z &\geq \sum_{e \in E} x_e w_e && && (22)\\
z &\geq -\sum_{e \in E} x_e w_e && && (23)\\
x_e &\in \{0,1\} && \forall e \in E \\
z &\in \mathbb{Z}
\end{aligned}
$$

Constraint (20) ensures the number of selected edges to be $n-1$. Constraints (21) ensures no cycle in the selected edges. Constraints (22) and (23) forces the objective function to be the balanced sum of selected edges.

2.5 Numerical results on sub-cases

As for the balanced TSP, results were obtained on an Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz core with 8 GB RAM and Linux Mint 11. The MIP models were implemented using Gurobi 8.1 and python2.7. Tables 2, 3, 4 and 5 present numerical results for respectively the Balanced assignment problem, the balanced spanning tree problem, the balanced subset problem and the TSP.

For the balanced assignment problem and the balanced subset problem, the result tables are similar to the balanced TSP (*i.e.* time to optimal with the lifted formulation and the bounds and gap for the standard formulation). For the classical TSP, since both formulations were able to prove optimality within a few seconds, we only use the time to optimal for both formulations. Finally, for the balanced spanning tree, since both formulations struggle to find optimality proofs for some instances, we stopped them if they run more than 30 seconds. We report the bounds of both formulations.

We note that for both the balanced assignment and the balanced subset, we obtain similar results as for the balanced TSP. For the classical TSP, we do not notice any significant difference between the two formulations. More tests should be performed on different non-balanced problems to evaluate the efficiency (or inefficiency) of the lifted formulation. Finally, the balanced minimum spanning tree is harder to solve than the other versions (including the balanced TSP). Indeed, even for some very small instances ($n = 40$), the lifted formulation is not able to prove optimality within 30 seconds. We note that the biggest solved instance within 30 seconds with the lifted formulation has 300 vertices. Also, the standard formulation is only able to solve the smallest instance ($n = 10$). Even for $n = 15$, the standard formulation is not able to find an optimal lower bound nor an optimal upper bound.

3 Conclusion & Perspectives

This paper presents a new MIP model for the balanced TSP. This model uses a specific property on the edge weights. This allows to make a clever reasoning and closing instances of the Metaheuristic Summer School (MESS18) competition. This paper investigates this phenomenon on several variants of the balanced TSP (namely the balanced assignment problem, the balanced fixed subset problem, the balanced minimum spanning tree problem and the classical TSP). We show that this performance improvement occurs on all balanced problems considered. However, this phenomenon seems to not appear on the classical TSP. We showed a dramatic performance increase on the lifted formulation compared to the standard formulation. This approach seems to be suited for balanced problems and a simple MIP model is even able to compete with other metaheuristics presented during the competition.

MIPs are known to be suited for instances of reasonable size. However, this approach is less suited on bigger instances (*i.e.* 5.000, 10.000, *etc.*). One way to overcome this issue is to implement *matheuristics*. One can, for instance, implement some Large Neighbourhood Search while using a MIP solver ([FL03]). This, way, one can take advantage of the progress made by such software and the lifted formulation described in the present paper.

References

FL03. Matteo Fischetti and Andrea Lodi. Local branching. *Mathematical programming*, 98(1-3):23–47, 2003.

A Appendix

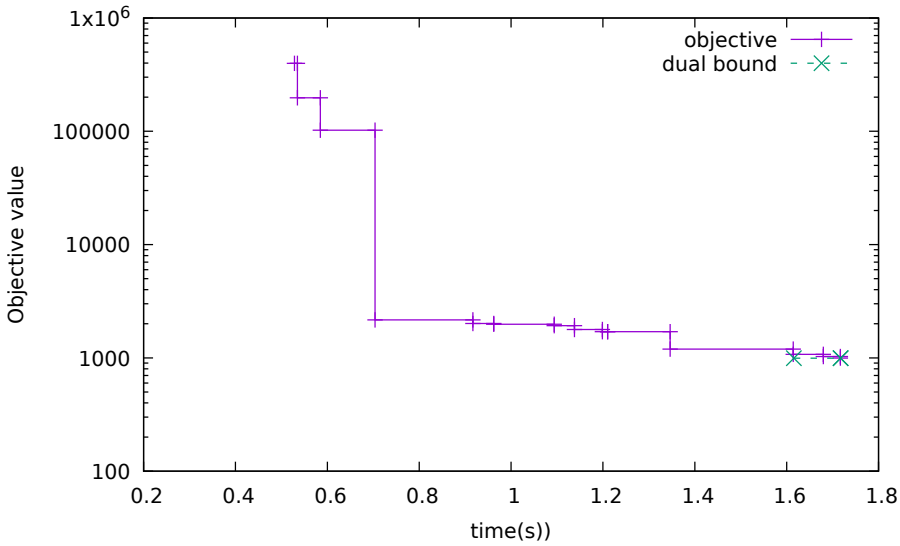

Fig. 1: btsp 0100 convergence curves

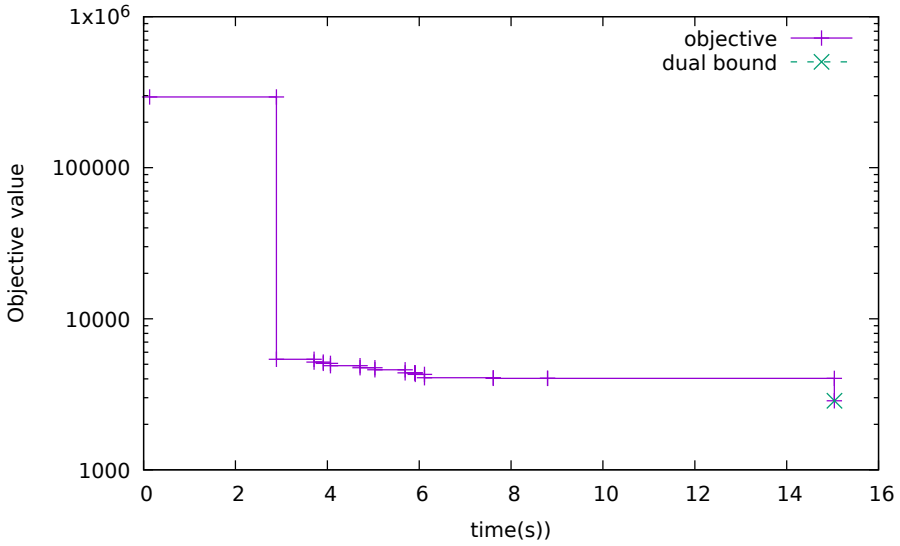

Fig. 2: btsp 0250 convergence curves

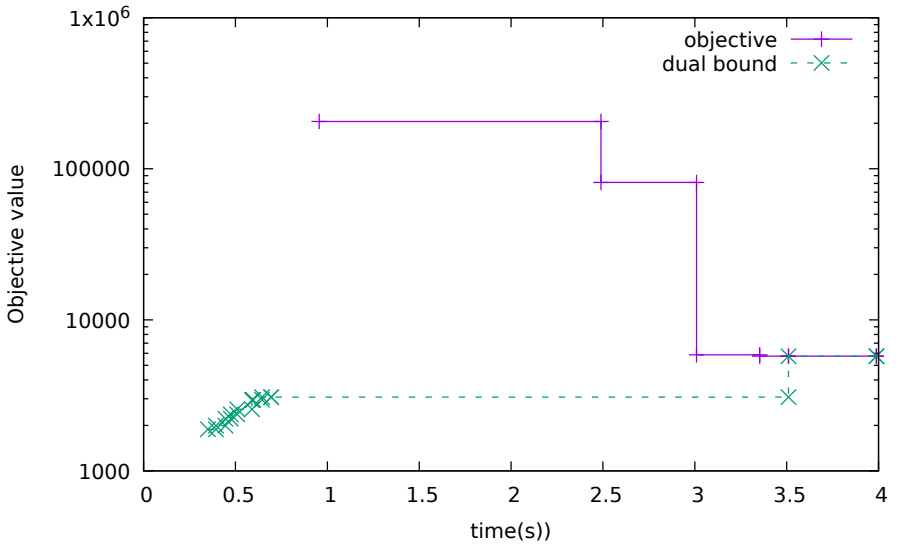

Fig. 3: btsp 0500 convergence curves

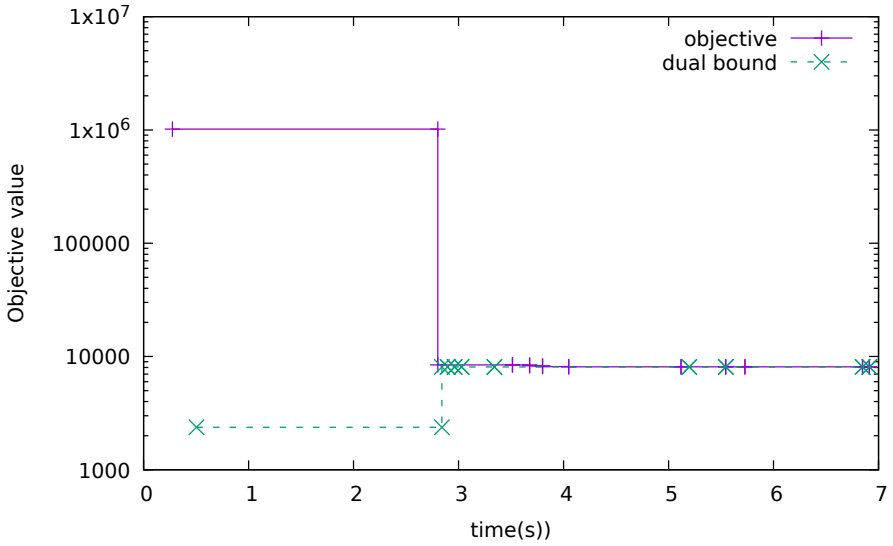

Fig. 4: btsp 0700 convergence curves

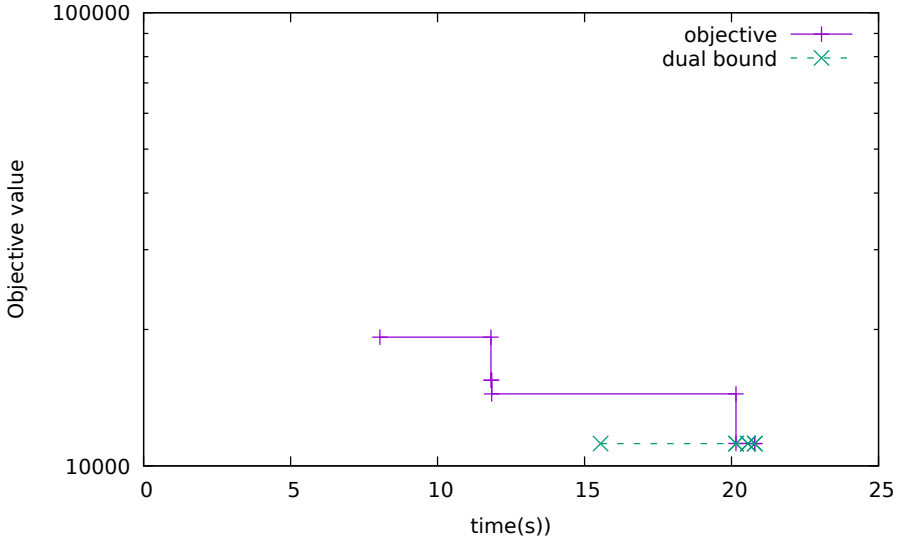

Fig. 5: btsp 1000 convergence curves

instance	time to optimal (s)	optimal objective	lower bound standard	upper bound standard	gap standard (%)
0010	0	105	105	105	0.0
0015	0	271	271	271	0.0
0020	0	285	221	286	22.7
0025	0	364	214	375	42.9
0030	0	433	161	443	63.6
0040	0	458	56	457	87.7
0050	0	629	117	805	85.4
0060	0	821	50	1.083	95.3
0070	0	857	90	1.210	92.5
0080	0	806	225	850	73.5
0090	0	973	58	1.203	95.1
0100	0	993	8	1.399	100.0
0150	0	1.664	0	2.868	100.0
0200	0	2.029	1	3.936	100.0
0250	0	2.796	0	5.056	100.0
0300	0	3.693	141	6.721	97.5
0400	1	4.698	0	7.099	100.0
0500	1	5.737	0	8.922	100.0
0600	0	6.543	4	9.789	99.9
0700	1	8.095	0	13.028	100.0
0800	1	9.226	0	14.899	100.0
0900	2	9.265	0	15.412	100.0
1000	2	11.201	0	21.157	100.0
1500	16	16.337	0	31.883	100.0
2000	18	20.755	0	47.645	100.0
2500	39	1.326	0	31.943	100.0
3000	38	0	0	18.872	100.0

Table 2: Numerical results on the balanced assignment problem

instance	lited lower bound	lifted upper bound	standard lower bound	standard upper bound
0010	60	60	60	60
0015	129	129	4	173
0020	143	143	0	242
0025	222	222	0	430
0030	247	247	0	247
0040	241	840	0	720
0050	378	378	0	1.036
0060	389	394	0	1.071
0070	375	375	0	1.238
0080	369	369	0	1.167
0090	342	342	0	1.300
0100	525	525	0	2.313
0150	855	855	0	2.476
0200	925	925	0	4.311
0250	1.170	1.170	0	3.236
0300	1.471	1.471	0	4.854
0400	2.099	2.109	0	7.493
0500	2.619	2.629	0	6.942
0600	0	2.917	0	10.381
0700	0	3.864	0	15.749
0800	0	4.319	0	15.649
0900	325	4.189	0	18.736
1000	386	5.188	0	21.090
1500	0	20.396	0	32.876
2000	0	23.264	0	45.489
2500	0	24.583	0	30.485
3000	0	28.297	0	13.466

Table 3: Numerical results on the balanced spanning-tree problem

instance	time to optimal (s)	optimal objective	lower bound standard	upper bound standard	gap standard (%)
0010	0	82	0	82	100.0
0015	0	130	0	130	100.0
0020	0	144	92	332	72.2
0025	0	233	0	306	100.0
0030	0	259	0	434	100.0
0040	0	263	0	460	100.0
0050	0	389	0	700	100.0
0060	0	406	128	568	77.4
0070	0	376	0	650	100.0
0080	0	358	0	1.750	100.0
0090	0	353	0	486	100.0
0100	0	537	0	2.074	100.0
0150	0	799	0	3.519	100.0
0200	0	926	0	4.176	100.0
0250	0	1.192	0	5.032	100.0
0300	0	1.493	0	5.785	100.0
0400	0	2.110	0	8.620	100.0
0500	0	2.630	0	10.708	100.0
0600	0	2.861	0	12.612	100.0
0700	1	3.886	0	14.463	100.0
0800	0	4.320	0	15.765	100.0
0900	0	4.166	0	16.042	100.0
1000	9	5.210	0	21.334	100.0
1500	19	7.579	0	33.974	100.0
2000	34	8.447	0	41.608	100.0
2500	7	0	0	29.713	100.0
3000	0	0	0	16.087	100.0

Table 4: Numerical results on the balanced subset problem

instance	time to optimal (s)	standard time to optimal (s)
0010	0	0
0015	0	0
0020	0	0
0025	0	0
0030	0	0
0040	0	0
0050	0	0
0060	0	0
0070	0	0
0080	0	0
0090	0	0
0100	0	0
0150	0	0
0200	0	0
0250	0	0
0300	0	0
0400	0	0
0500	0	0
0600	0	0
0700	0	0
0800	0	0
0900	0	0
1000	1	1
1500	0	0
2000	2	2
2500	4	3
3000	2	2

Table 5: Numerical results on the classical TSP problem

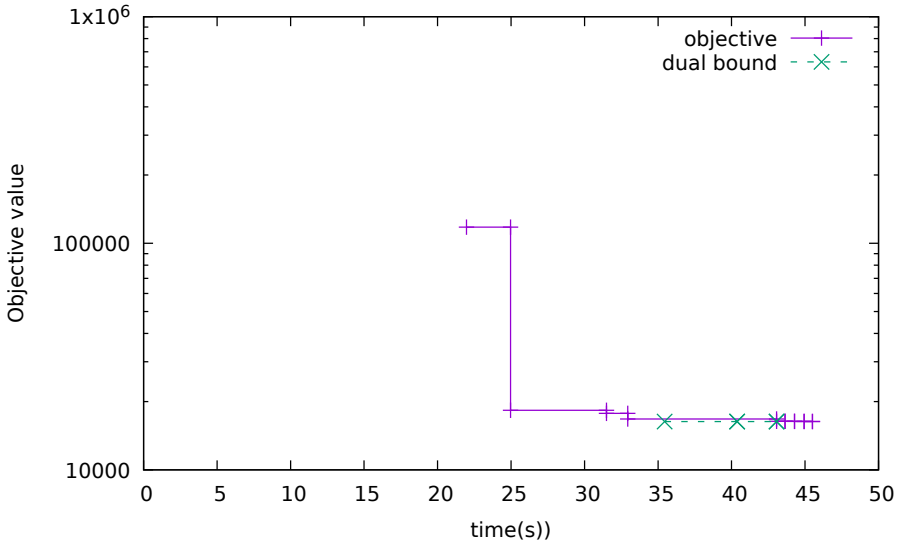

Fig. 6: btsp 1500 convergence curves