Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

TOBIAS ACHTERBERG
THORSTEN KOCH
ANDREAS TUCHSCHERER

On the Effects of Minor Changes in Model Formulations

supported by the DGF Research Center Matheon

ZIB-Report 08-29 (August 2008)

On the Effects of Minor Changes in Model Formulations*

Tobias Achterberg† Thorsten Koch‡ Andreas Tuchscherer§

Abstract

Starting with the description of the Traveling Salesmen Problem formulation as given by van Vyve and Wolsey in the article "Approximate extended formulations", we investigate the effects of small variations onto the performance of contemporary mixed integer programming solvers. We will show that even minor changes in the formulation of the model can result in performance difference of more than a factor of 1000. As the results show it is not obvious which changes will result in performance improvements and which not.

1 Introduction

In their article [vVW06] "Approximate extended formulations" van Vyve and Wolsey describe a mixed integer programming (MIP) model for solving the Traveling Salesmen Problem (TSP). When we tried to reproduce the results we noticed an erratic behavior of the MIP solvers depending on minor variations of the model. In the following we will show that even very small changes in the formulation can have a large impact on the solvability. Trick shows in [Tri05] that with modern MIP solvers the effects of changes are hard to predict as today's solvers employ many methods to apply the usual tricks used to improve a formulation. In this article we will show that probably due to these automatic "improvements" the performance can get nearly unpredictable. In particular, omitting seemingly useful redundant information can sometimes speed up the solution process.

In the next section we will present the model as described in the original article and list possible variations. In Section 3 computational results for all combinations of the variations will be given and explained.

2 The model and its variations

The TSP instance is defined by a set of nodes $V = \{1, \dots, n\}$, $n \in \mathbb{N}$, a set of arcs $A = \{(i,j) | i, j \in V, i \neq j\}$, and a distance (weight, cost) function c_{ij}, interpreted as the length of arc $(i,j) \in A$. The goal is to find a shortest round trip through all nodes $i \in V$.

2.1 The formulation by van Vyve and Wolsey

The formulation depends on an approximation parameter k which controls the extend of the subtour elimination constraints implied by the model. For each node $l \in V$, define a neighborhood $V_l \subseteq V$ as the set of the k nodes nearest to node l, including l itself. The model is given in terms

*Supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.

†ILOG Deutschland GmbH, tachterberg@ilog.de

‡Zuse Institute Berlin (ZIB), Takustr. 7, 14195 Berlin, Germany, koch@zib.de

§Zuse Instritute Berlin, tuchscherer@zib.de

of the following variables:

$$y_{ij} := \begin{cases} 1, & \text{if arc } (i,j) \in A \text{ is in the tour,} \\ 0, & \text{otherwise.} \end{cases}$$

$$u_i := \text{number of nodes visited before } i \in V \text{ (node 1 is visited first)}$$

$$w_{ij}^l := \text{flow on arc } (i,j) \in A \text{ for neighborhood } l \in V$$

Now the MIP formulation for the Traveling Salesman Problem reads as follows:

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij} \quad \text{s. t.} \tag{1}$$

$$\sum_{(j,i)\in\delta^-(i)} y_{ji} = 1 \quad \text{for all } i \in V \tag{2}$$

$$\sum_{(i,j)\in\delta^+(i)} y_{ij} = 1 \quad \text{for all } i \in V \tag{3}$$

$$y_{ij} \in \{0,1\} \quad \text{for all } (i,j) \in A \tag{4}$$

$$u_1 = 0 \tag{5}$$

$$u_i - u_j + (n-1)y_{ij} \le n-2 \quad \text{for all } (i,j) \in A : j \ne 1 \tag{6}$$

$$u_i \ge 1 \quad \text{for all } i \in \{2,\dots,n\} \tag{7}$$

$$u_i \le n-1 \quad \text{for all } i \in \{2,\dots,n\} \tag{8}$$

$$\sum_{(i,j)\in\delta^-(j)} w_{ij}^l - \sum_{(j,i)\in\delta^+(j)} w_{ji}^l = 0 \quad \text{for all } l \in V \text{ for all } j \in V_l : j \ne l \tag{9}$$

$$\sum_{(i,l)\in\delta^-(l)} w_{il}^l - \sum_{(l,i)\in\delta^+(l)} w_{li}^l = 1 \quad \text{for all } l \in V \tag{10}$$

$$w_{ij}^l \ge 0 \quad \text{for all } l \in V,\ (i,j) \in A(V_l) \cup \delta^-(V_l) \cup \delta^+(V_l) \tag{11}$$

$$w_{ij}^l \le y_{ij} \quad \text{for all } l \in V,\ (i,j) \in A(V_l) \cup \delta^-(V_l) \tag{12}$$

Note that already the model (1)–(8) is a correct TSP formulation (Miller-Tucker-Zemlin formulation [PS91]). Constraints (9)–(12) are useful as they imply subtour elimination constraints as specified by

Theorem 1 (van Vyve, Wolsey)**.** *The subtour elimination constraint*

$$\sum_{(i,j)\in\delta^-(U)} y_{ij} \ge 1$$

is valid for (9)–(12) *if there exists* $l \in V$ *such that* $l \in U \subseteq V_l$*.*

Proof. Let l and U satisfying $l \in U \subseteq V_l$ be given. Equations (9)–(12) guarantee that there is enough capacity in y for one unit to flow from outside V_l to l. Thus the capacity of the cut $\delta^-(U)$ is at least 1. □

So far we have ignored one important issue: In case $k = n$, the model is infeasible since constraints (9)–(12) cannot be satisfied. In order to maintain feasibility, in the original formulation the neighborhood sets do not contain node 1, i. e., V_l is defined as the set of the k nodes nearest to node l, including l itself, but excluding node 1. However, removing node 1 from the sets V_l, for $l \in V$, degrades the solvability of the model for $k < n$ as it significantly reduces the initial lower bound. Since the whole point in using *approximate* extended formulations is to use $k < n$, we include node 1 in all computations.

2.2 Possible variations of the formulation

In the following we list some minor changes of the formulation. Regarding the model the restrictions are redundant and the relaxations only remove redundant constraints. In all cases neither the integer optimal solution is altered, nor the objective value of the linear programming relaxation is changed.

2.2.1 Restricting the formulation

1. The variables u_i have to be integer in any feasible solution.

$$u_i \in \{1, \ldots, n-1\} \quad \text{for all } i \in V \tag{13}$$

can be added to the model.

2. There are no explicit upper bounds on the variables w_{ij}^l, even though all are implicitly bounded by inequality (12):

$$0 \leq w_{ij}^l \leq 1 \quad \text{for all } l \in V_l, (i,j) \in A(V_l) \cup \delta^-(V_l) \cup \delta^+(V_l) \tag{14}$$

3. The variables w_{ij}^l are also implicitly integer.

$$w_{ij}^l \in \mathbb{Z} \quad \text{for all } l \in V_l, (i,j) \in A(V_l) \cup \delta^-(V_l) \cup \delta^+(V_l) \tag{15}$$

can be added to the model.

4. All w_{ij}^l in $(i,j) \in \delta^+(V_l)$ can be fixed to zero:

$$w_{ij}^l = 0 \quad \text{for all } l \in V_l, (i,j) \in \delta^+(V_l) \tag{16}$$

5. For the same reason upper bounds on w_{ij}^l in $(i,j) \in \delta^+(V_l)$ can be added in inequality (12), using instead:

$$w_{ij}^l \leq y_{ij} \quad \text{for all } l \in V,\ (i,j) \in A(V_l) \cup \delta^-(V_l) \cup \delta^+(V_l) \tag{17}$$

2.2.2 Relaxing the formulation

6. The upper bounds on the u_i variables are not needed, constraint (8) can be omitted.

7. For equations (9) and (10) equality is not required. Relaxing them to

$$\sum_{(i,j)\in\delta^-(j)} w_{ij}^l - \sum_{(j,i)\in\delta^+(j)} w_{ji}^l \geq 0 \quad \text{for all } l \in V \text{ for all } j \in V_l : j \neq l \tag{18}$$

$$\sum_{(i,l)\in\delta^-(l)} w_{il}^l - \sum_{(l,i)\in\delta^+(l)} w_{li}^l \geq 1 \quad \text{for all } l \in V \tag{19}$$

is still valid.

3 Computational results

In this section we will report on the computational results obtained by solving all combinations of the above variations of one TSP instance with two contemporary MIP solvers.

All computing times are given as CPU seconds on a PC running Linux with a 3.6 GHz Pentium-D and 4 GB RAM. If not otherwise noted, all runs are limited to at most one hour. The instance used is *att48* from TSPLIB [Rei91]. $k = 13$ is used in all cases. The model was generated using ZIMPL[1] [Koc04] version 2.07. The source files can be found at http://www.zib.de/koch/reformulation. We used CPLEX[2] version 10.0.1 and SCIP[3] [Ach04] version 0.90e to solve the MIP instances. CPLEX and SCIP were run with default settings, with the exception that probing was disabled for SCIP (in CPLEX, only a very limited version of probing is applied by default). SCIP used CPLEX 10.0.1 as LP solver subroutine.

The formulation variants are denoted as *u-w-e-b-f*. Table 1 describes possible values and their meanings. The original formulation is denoted as *2-1-1-1-0*. Note that $u = 3$ and $w = 3$ mean the variable is an *implicit integer*, i. e., the variable will have an integral value for any solution of the LP relaxation where all normal integer variables have integral values. With ZIMPL and SCIP it is possible to notify the solver of this property. The solver has then the possibility to treat the variable alternatively as continuous or integer variable.

u		Description
1	$1 \leq u_i \leq \infty$	Relaxation 6 used (Constraint (8) omitted)
2	$1 \leq u_i \leq n-1$	as in the original formulation
3	$1 \leq u_i \leq n-1$	but declared as *implied integer* (SCIP only)
4	$u_i \in \{1,\ldots,n-1\}$	Restriction 1 (declared integer)
5	$u_i \in \mathbb{N}$	Restriction 1 and Relaxation 6
w		
1	$0 \leq w_{ij}^l \leq \infty$	as in the original formulation
2	$0 \leq w_{ij}^l \leq 1$	Restriction 2 (explicit upper bounds)
3	$0 \leq w_{ij}^l \leq 1$	Restriction 2 and declared as *implicit integer* (SCIP only)
4	$w_{ij}^l \in \{0,1\}$	Restriction 2 and 3 (declared binary)
5	$w_{ij}^l \in \mathbb{N}$	Restriction 3 (declared integer)
e		
0		Relaxation 7 (not requiring equality for (9) and (10))
1		as in the original formulation
b		
0		Restriction 5 (bind outgoing w_{ij}^l to y_{ij})
1		as in the original formulation
f		
0		as in the original formulation
1		Restriction 4 (fix outgoing w_{ij}^l to zero)

Table 1: Possible values of formulation variants denoted as *u-w-e-b-f*.

[1] http://zimpl.zib.de
[2] http://www.ilog.com/cplex
[3] http://scip.zib.de

3.1 Comments and Equivalent Settings

In case $f = 1$, the setting for b is irrelevant. Therefore, both parameters are better combined into a single. A more suitable parameter scheme is maybe the following:

i		Description
0		as in the original formulation
1		Relaxation 7 (not requiring equality for (9) and (10))
b		
0		as in the original formulation
1		Restriction 4 (fix outgoing w^l_{ij} to zero)
2		Restriction 5 (bind outgoing w^l_{ij} to y_{ij})
w		
1	$0 \le w^l_{ij} \le \infty$	as in the original formulation
2	$0 \le w^l_{ij} \le 1$	Restriction 2 (explicit upper bounds)
3	$0 \le w^l_{ij} \le 1$	Restriction 2 and declared as *implicit integer* (SCIP only)
4	$w^l_{ij} \in \{0,1\}$	Restriction 2 and 3 (declared binary)
5	$w^l_{ij} \in \mathbb{N}$	Restriction 3 (declared integer)
u		
1	$1 \le u_i \le \infty$	Relaxation 6 used (Constraint (8) omitted)
2	$1 \le u_i \le n-1$	as in the original formulation
3	$1 \le u_i \le n-1$	but declared as *implied integer* (SCIP only)
4	$u_i \in \{1,\dots,n-1\}$	Restriction 1 (declared integer)
5	$u_i \in \mathbb{N}$	Restriction 1 and Relaxation 6

Table 2: Possible values of formulation variants denoted as *i-b-w-u*.

The following situations may yield identical models after preprocessing up to the ordering of some constraints (f-values indicate arbitrarily fixed parameters):

- $i = 1, b = z \in \{0,1,2\}, w = f_1, u = f_2$: Here the choice of z is irrelevant since outgoing w^l_{ij} can be fixed to zero in preprocessing in all cases.
- $i = 1, b \in \{0,1,2\}, w = f_1 \in \{1,5\}, u = f_2 \iff i = 0, b = 0, w = f_1, u = f_2$: In all these situation the effective presprocessing is possible.
- $(i = 1)$ or $(i = 0, b \in \{1,2\})$, i. e., $\neg(i = 0, b = 0)$: Independent of the choice of u (and b in the first case), we have $w = 1 \iff w = 2$ and $w = 4 \iff w = 5$.

3.2 Preprocessing

As can be expected with such minor formulation differences after preprocessing basically only three mixed integer programs remain, regarding number of columns (variables), rows (constraints), and number of non-zero entries in the constraint matrix. Details are shown in Table 3.

Most reductions are made in preprocessing if the constraints (9) and (10) are formulated using inequalities. This can also be achieved in the setting *u-1-1-1-0*, where the variables w^l_{ij} for $(i,j) \in \delta^+(V_l)$ have no upper bounds and thus correspond to slack in (9) and (10). Removing these variables yields also inequalities in these constraints. Having inequalities preprocessing works as follows: Variables w^l_{ij} for $l \in V$ and $(i,j) \in \delta^+(V_l)$ can be fixed to zero since each such variable appears only in one constraint and has objective zero (column singleton). Moreover, the tightest upper bound on each variable w^l_{ij} for $(i,j) \in \delta^-(V_l)$ is given implicitly by y_{ij}. Therefore,

	Size	Cols	Rows	Non-zeros	
S	Small	9,791	10,370	62,790	CPLEX & SCIP
M1	Medium 1	31,631	32,210	106,470	CPLEX & SCIP
M2	Medium 2	32,255	32,210	107,094	CPLEX
L1	Large 1	52,847	32,210	127,686	SCIP
L2	Large 2	53,461	54,050	171,990	CPLEX & SCIP

Table 3: Instance sizes after preprocessing

variable w_{ij}^l can be substituted by y_{ij}. Altogether, only the variables w_{ij}^l for $(i,j) \in A(V_l)$ remain after preprocessing, reducing the number of variables w_{ij}^l from $n(k(k-1)+2k(n-k))$ to $nk(k-1)$.

The worst case concerning preprocessing is *u-w-1-0-0*, where particularly the variables w_{ij}^l for $(i,j) \in \delta^+(V_l)$ are bounded by y_{ij}. In this situation no reductions on the model are possible at all since there are no more any column singletons with zero objective. Moreover, no variable can be fixed at one of its bounds.

3.3 CPLEX

We investigated all 128 formulation variants. It turned out that if $e = 0$, i. e., equality is not required in constraints (9) and (10), it makes no difference whether the outgoing w_{ij}^l variables are fixed to zero ($f = 1$) or not ($f = 0$). Given the above described preprocessing results, this is not remarkable. On the other hand, if we require equality in (9) and (10) any combination of settings for b and f gives different results, i. e., solving needs a different number of branch-and-cut nodes or at least a different number of total simplex iterations. For the remaining 96 instances for any setting which is different from *u-w-1-1-0* $w = 1$ is equal to $w = 2$, and $w = 4$ is equal to $w = 5$. The reason for this is that the preprocessor is able to deduce the implicitly given bounds on the w_{ij}^l variables. For those of the remaining 56 instances of type *u-1-e-1-0* the setting of e does not matter. This leaves 52 different instances altogether.

The results for these instances are shown in Table 5. The first column list the variation used, column *Size* gives the resulting size of the instance after preprocessing according to Table 3. *Iters* is the total number of Simplex iterations, *Nodes* is the number of branch-and-cut nodes processed and *time* list the solution time in seconds. The optimal objective value is 10628, the objective value of the root relaxation before any cuts is always 10604.

The fastest formulation is *1-4-0-0-0* which is solved by CPLEX in 27 seconds. On the other hand, *5-1-0-1-0*, even though of the same small size can not be solved to optimality in 24 hours. The only difference between these formulations is that in case of *1-4-0-0-0* the u_i variables are continuous and the w_{ij}^l variables are binary, while in case of *5-1-0-1-0* the u_i are integer and the w_{ij}^l are continuous variables with upper bounds. This small difference is enough to result in a factor of more than 3200 for the required solution time.

Table 4 shows the geometric mean over all results with a particular parameter setting. The column labeled *Nodes* contains the mean number of branch-and-bound nodes required over all instance where the parameter given in the *Setting* column had the listed value. Column *Time* list the mean required solution time in seconds. The large difference in the number of nodes between CPLEX and SCIP is due to different default branching strategies. A detailed explanation can be found in [AKM05].

As can be seen, using $u = 1$ and $e = 0$ are particular winners. Setting the variables to continuous instead of integer performs better in general. The setting of b and f seems only to matter in relation to the resulting size of the instance after preprocessing.

From the average results we might guess that *1-2-0-1-0* is the fastest setting (given as *1-1-0-1-0* by similarity in Table 5). With 60 seconds the actual performance is more than twice as slow as the actual winner, but still considerably faster than the overall geometric mean of 244 seconds.

Setting	Nodes CPLEX	Nodes SCIP	Time [s] CPLEX	Time [s] SCIP
Total	666	62	243.8	244.9
$u=1$	146	48	85.5	238.4
$u=2$	427	48	142.0	220.4
$u=3$		43		224.5
$u=4$	1,396	88	440.9	266.2
$u=5$	2,259	106	661.1	280.5
$w=1$	1,211	69	222.7	221.0
$w=2$	790	63	203.2	236.3
$w=3$		43		242.3
$w=4$	513	69	299.5	281.0
$w=5$	400	73	260.9	247.6
$e=0$	972	46	118.7	137.1
$e=1$	456	84	500.7	437.3
$b=0$	571	78	260.7	314.7
$b=1$	776	50	228.1	190.5
$f=0$	584	74	205.5	260.3
$f=1$	759	52	289.3	230.4

Table 4: Geometric mean of number of branch-and-bound nodes and solution time for all instances with a particular setting

3.4 SCIP

Again for SCIP $w = 1$ is equal to $w = 2$, and $w = 4$ is equal to $w = 5$ for any setting which is different from *u-w-1-1-0*. All remaining 130 settings lead to a different solver run, as can be seen in Table 6. The column labeled *rs* depicts the number of *restarts*. This feature of SCIP is triggered by fixings found in the root node. It reinvokes the presolving procedure to clean up the model and deduce further reductions. It is noteworthy that SCIP was able to solve all instances within an hour and that the ratio between the slowest and the fastest run is only 36 compared to over 3000 for CPLEX. The reason is most probably the more expensive branching strategy of SCIP which involves more strong branching and thus takes much longer per node. As a result CPLEX is nearly three times faster for the best run. If we look at average results for SCIP in Table 4, we see that $e = 0$ is again a clear winner. The setting of b has much more influence in SCIP than in CPLEX. Contrary to CPLEX fixing outgoing variables ($f = 1$) is advantageous for SCIP. The u and w settings, i. e., the type of variables does not matter that much in SCIP, even though declaring the variables integer is again slightly inferior. From the average results, one would expect *2-1-0-1-1* to be the winning combination, so both solvers work best on continuous variables.

We expected the implicit integer declaration ($u = 3$, $w = 3$) to perform at least as good as the bounded continuous settings ($u = 2$, $w = 2$). Unfortunately, this is not the case. This suggests that the current implementation of implied integers in SCIP is not able to exploit all advantages of this information.

4 Conclusion

The obvious conclusion from the results in this article is that it is very important to precisely state the model formulation when reporting computational results. Otherwise reproducibility will be difficult to achieve.

Furthermore, it becomes evident that the intractability of a specific formulation of a model

u-w-e-b-f	Size	Iters	Nodes	Time
1-4-0-0-0	S	7410	37	27
1-4-0-1-0	S	10481	59	35
2-4-0-1-0	S	14692	130	40
2-2-1-1-0	M2	35414	48	44
1-1-1-1-1	M1	41234	86	57
1-1-0-1-0	S	28467	418	60
2-1-1-1-1	M1	41130	56	60
1-2-1-1-0	M2	43684	89	66
2-1-0-0-0	S	41912	961	68
1-1-0-0-0	S	50642	775	71
2-1-1-0-1	M1	42028	95	77
4-1-0-0-0	S	54047	781	77
5-1-0-0-0	S	59458	1013	81
1-1-1-0-1	M1	37753	70	86
1-1-1-0-0	L2	41883	57	89
2-1-1-0-0	L2	45956	96	89
2-1-0-1-0	S	87268	1851	90
5-5-1-1-0	M2	13278	40	114
4-2-1-1-0	M2	79603	126	129
1-4-1-1-1	M1	45113	41	137
5-4-1-1-1	M1	42272	39	138
2-4-0-0-0	S	68172	1090	142
4-5-1-1-0	M2	17662	42	143
4-1-0-1-0	S	109418	3015	165
4-4-0-1-0	S	89950	1238	175
1-4-1-0-0	L2	44654	33	196
5-4-0-0-0	S	106546	1429	198
2-4-1-0-0	L2	53772	41	238
4-4-0-0-0	S	147740	1667	240
1-5-1-1-0	M2	53416	349	288
1-4-1-1-0	M2	103406	805	325
5-4-0-1-0	S	253199	3559	358
5-2-1-1-0	M2	209979	710	462
2-4-1-1-0	M2	152528	723	511
2-5-1-1-0	M2	116394	860	580
5-4-1-0-0	L2	154083	246	695
4-1-1-1-1	M1	784910	1864	1004
5-4-1-1-0	M2	341919	1314	1046
4-4-1-1-0	M2	412584	2015	1279
4-4-1-0-1	M1	747781	1277	1676
1-4-1-0-1	M1	724116	1720	2218
4-4-1-0-0	L2	585662	1474	2265
2-4-1-1-1	M1	822014	1908	2400
4-1-1-0-0	L2	666077	1189	2450
5-1-1-0-0	L2	972289	1629	2546
5-4-1-0-1	M1	1033004	2055	2637
2-4-1-0-1	M1	1007944	2073	2802
5-1-1-0-1	M1	1694751	3133	2993
4-4-1-1-1	M1	1340441	2385	3391
4-1-1-0-1	M1	$>$ 2316820	$>$ 4912	$>$ 3600
5-1-0-1-0	S	$>$ 4791738	$>$ 138736	$>$ 3600
5-1-1-1-1	M1	$>$ 2234432	$>$ 3653	$>$ 3600

Table 5: Results for CPLEX

u-w-e-b-f	Size	RS	Nodes	Time
4-1-0-1-1	S	1	14	79
1-3-0-1-0	S	1	15	90
1-3-0-0-0	S	1	10	93
1-1-1-1-0	S	1	14	96
3-1-0-1-1	S	3	14	97
2-3-0-1-1	S	1	12	98
5-3-0-1-0	S	1	21	98
5-4-0-0-1	S	1	19	99
1-1-0-0-1	S	4	11	101
1-3-0-1-1	S	1	15	104
4-5-1-1-0	S	1	15	104
5-1-1-1-0	S	1	13	104
1-1-0-1-0	S	3	12	106
4-3-0-1-0	S	1	14	106
3-3-0-1-0	S	3	17	107
5-1-0-1-1	S	1	25	107
2-1-0-1-1	S	5	29	110
2-1-1-1-0	S	3	107	110
2-3-0-0-1	S	1	10	110
1-3-0-0-1	S	1	75	113
1-5-1-1-0	S	1	24	113
2-1-0-1-0	S	5	61	113
3-3-0-1-1	S	1	21	114
4-4-0-0-1	S	1	40	114
1-4-0-0-1	S	1	14	115
5-4-0-1-1	S	1	17	115
3-1-0-0-0	S	7	12	118
4-3-0-0-0	S	1	17	120
3-4-0-1-0	S	1	15	123
3-4-0-1-1	S	1	58	123
4-3-0-0-1	S	1	28	124
4-4-0-0-0	S	1	80	125
2-3-0-0-0	S	7	22	127
2-1-0-0-1	S	6	15	128
3-3-0-0-1	S	1	38	128
5-1-0-0-1	S	1	67	130
5-3-0-1-1	S	1	124	131
4-3-0-1-1	S	1	22	132
3-4-0-0-1	S	1	13	133
1-4-0-1-0	S	1	82	136
3-4-0-0-0	S	1	66	137
4-4-0-1-1	S	1	19	139
5-4-0-1-0	S	1	127	140
2-3-0-1-0	S	1	22	142
2-5-1-1-0	S	1	25	142
1-4-0-0-0	S	1	76	143
4-1-1-1-0	S	1	80	150
2-4-0-1-0	S	1	134	155
1-4-0-1-1	S	1	59	156
2-1-0-0-0	S	1	55	156
2-4-0-1-1	S	1	57	156
3-1-0-0-1	S	5	25	158
2-4-0-0-1	S	1	86	160
5-1-0-1-0	S	1	229	161
3-1-1-1-1	M1	2	20	167
4-4-1-1-0	L1	1	9	167
4-1-0-1-0	S	1	199	168
1-1-0-1-1	S	3	37	169
4-1-0-0-1	S	1	62	169
3-1-1-1-0	S	1	64	171
1-1-0-0-0	S	1	66	177
1-2-1-1-0	L1	2	13	179
3-3-1-1-0	L1	3	22	182
2-1-1-0-1	M1	2	17	184
3-3-0-0-0	S	1	49	184
4-4-0-1-0	S	1	120	185
2-1-1-1-1	M1	2	12	188
2-2-1-1-0	L1	3	17	188
1-1-1-0-1	M1	3	118	191
3-1-1-0-1	M1	2	28	191
3-2-1-1-0	L1	2	25	191
5-5-1-1-0	S	1	116	199
2-4-0-0-0	S	1	121	200
5-3-0-0-1	S	1	100	200
3-5-1-1-0	S	1	96	201
3-1-0-1-0	S	1	224	207
5-3-0-0-0	S	1	225	207
5-4-0-0-0	S	1	102	210
4-1-0-0-0	S	1	338	223
4-2-1-1-0	L1	1	12	228
1-3-1-1-0	L1	1	13	232
2-3-1-1-0	L1	3	51	248
5-2-1-1-0	L1	1	47	270
3-4-1-1-0	L1	1	47	280
5-1-0-0-0	S	1	875	301
4-3-1-1-0	L1	1	13	306
2-4-1-1-0	L1	1	127	319
5-3-1-1-0	L1	1	222	350
5-4-1-1-0	L1	1	154	376
5-1-1-0-1	M1	1	125	388
5-1-1-1-1	M1	1	196	395
4-3-1-1-1	M1	1	58	397
2-3-1-1-1	M1	1	68	412
1-4-1-1-0	L1	1	69	420
3-3-1-1-1	M1	1	75	426
4-1-1-1-1	M1	1	344	433
5-4-1-1-1	M1	1	129	437
3-3-1-0-1	M1	1	29	441
4-1-1-0-1	M1	1	301	441
1-4-1-1-1	M1	1	79	455
3-4-1-0-1	M1	1	66	461
2-4-1-1-1	M1	1	72	477
4-4-1-1-1	M1	1	88	489
1-1-1-1-1	M1	1	112	492
2-3-1-0-1	M1	1	107	507
5-3-1-1-1	M1	1	123	519
1-3-1-0-1	M1	1	106	522
1-3-1-0-0	L2	1	25	529
4-3-1-0-1	M1	1	95	602
5-3-1-0-1	M1	1	96	623
1-3-1-1-1	M1	1	217	664
3-4-1-1-1	M1	1	136	680
2-4-1-0-1	M1	1	76	687
2-4-1-0-0	L2	1	49	705
5-3-1-0-0	L2	1	148	737
4-4-1-0-0	L2	1	176	782
5-4-1-0-1	M1	1	184	810
3-4-1-0-0	L2	1	145	813
5-4-1-0-0	L2	1	116	837
2-3-1-0-0	L2	1	47	863
2-1-1-0-0	L2	1	198	875
1-4-1-0-1	M1	1	127	895
3-3-1-0-0	L2	1	78	916
3-1-1-0-0	L2	1	178	937
1-4-1-0-0	L2	1	91	984
4-3-1-0-0	L2	1	191	1075
1-1-1-0-0	L2	1	367	1420
4-4-1-0-1	M1	1	406	1548
4-1-1-0-0	L2	1	968	2088
5-1-1-0-0	L2	1	710	2879

Table 6: Results for SCIP

using a specific solver does not necessary imply that the model in general is intractable. Slight modifications to the formulation may have a big impact on solvability. Therefore, it might be useful to identify solving strategies that are likely to be independent of the specific formulation. One candidate in this regard clearly is the branching strategy. The strategy used in SCIP, while considerably slower in the best case, has a much smaller dependency on the formulation.

The basic problem is our inability to convey additional information to the solver which is redundant to the model but may give the solver more options to improve the solving process. As a first step, we have proposed the notion of *implicit integer* variables. This concept could be extended to the bounds of the variables and to the right hand sides and senses of the constraints. However, additional development is needed to reliably exploit this additional information in MIP solvers.

5 Acknowledgements

We would like to thank Laurence Wolsey for making the original Mosel model available to us.

References

[Ach04] Tobias Achterberg, *SCIP - a framework to integrate constraint and mixed integer programming*, Tech. Report 04-19, Zuse Institute Berlin, 2004, http://www.zib.de/Publications/abstracts/ZR-04-19/.

[AKM05] Tobias Achterberg, Thorsten Koch, and Alexander Martin, *Branching rules revisited*, Operations Research Letters **33** (2005), 42–54.

[Koc04] Thorsten Koch, *Rapid mathematical programming*, Ph.D. thesis, Technische Universität Berlin, 2004.

[PS91] Manfred Padberg and Ting-Yi Sung, *An analytical comparison of different formulations of the travelling salesman problem*, Mathematical Programming **52** (1991), 315–357.

[Rei91] Gerhard Reinelt, *TSPLIB – A Traveling Salesman problem library*, ORSA Journal on Computing **3** (1991), 376–384.

[Tri05] Michael Trick, *Formulations and reformulations in integer programming*, Lecture Notes in Computer Science **3524** (2005), 366–379.

[vVW06] Mathieu van Vyve and Laurence A. Wolsey, *Approximate extended formulations*, Mathematical Programming **105** (2006), 501–511.