Advanced analysis of branch and bound algorithms

Turkensteen, Marcel

Chapter 4

Additional Topics on Tolerance-Based Algorithms

4.1 Introduction

In Chapter 3, tolerance-based Branch and Bound (BnB) algorithms are introduced for the Asymmetric Traveling Salesman Problem (ATSP). The performance of these algorithms is compared to cost-based algorithms and BnB methods from the literature. Tolerance-based BnB methods prove to be effective for various instances of the ATSP, but not for all. In this chapter, we try to find instances for which tolerance-based BnB is most effective, and try to find reasons for it being so. Moreover, we suggest improvements to tolerance-based BnB algorithms, such as increased lower bounds.

First, we consider the theory of *complexity transitions*, explained in Hogg *et al.* (1996). The difficulty of a randomly generated instance of a COP often depends on the values of the parameters describing the distribution from which the random instance is drawn. Take for example the ATSP, introduced in Section 2.1. An instance of the ATSP is defined by the cost matrix C containing the distances between each pair of locations. Assume that each entry in the matrix C is drawn from a uniform distribution supported on $\{1, \ldots, R\}$. These randomly generated instances are more difficult to solve if the range of the distribution R is large; see Zhang and Korf (1996). In Section 4.2, we compare the performance of tolerance-based BnB algorithms with that of cost-based BnB for various values of R.

In Section 4.3, we consider alternative branching rules for the ATSP. In Section 3.5, we try to predict which arcs from an Assignment Problem (AP) solution are also in some optimal solution of the ATSP instance with the same cost matrix C. The results indicate that the predictions based on upper tolerance values are more accurate than those based on cost values. We restrict ourselves Depth First BnB algorithms. An inaccurate prediction at a node close to the root of the branch and bound tree would cause the algorithm to fruitlessly search through a large number of subproblems; see Section 3.5. It may be worthwhile to restrict the tolerance computations to a small part of the search tree, since these computations may take much time. Branching on tolerances only at top nodes of the search tree seems to make sense, but is this true? We compare *hybrid algorithms*, using tolerances at selected nodes and costs elsewhere, to the usual cost-based and tolerance-based algorithms from Section 3.7.

Finally, in Section 4.4, we try to improve tolerance-based lower bounds for problems that have the Minimum Spanning Tree Problem (MSTP) as a relaxation. The lower bounds from Section 3.6 are formed by adding the upper tolerance value of a single arc, even when there is a large number of subcycles that should be broken. When the number of offenders is large, one is inclined to remove multiple offenders simultaneously. We develop new lower bounds for the Degree-Constrained Minimum Spanning Tree Problem (DCMSTP) for which we use the regular Minimum Spanning Tree Problem (MSTP) as a relaxation. This lower bound is obtained by adding the upper tolerance values of specific edges.

4.2 Branch and Bound and Complexity Transitions

The fact that a problem is $\mathcal{NP}$-hard does not imply automatically that all very large instances of the problem are unsolvable. There exist conditions under which $\mathcal{NP}$-hard problem instances are actually polynomially solvable; see, for example, Burkard (1997). In this section, we consider problem instances that are *on average* polynomially solvable. This means the following. Suppose that instances of a COP are drawn from a certain distribution with prespecified parameter values. In case of the ATSP, we assume that the intercity distances in C are drawn from a uniform distribution supported on $\{1, \ldots, 1000\}$. In Section 1.3,

a definition of average case complexity is given. What is, on average, the complexity of such randomly generated ATSP instances?

In this section, we compare two algorithms from Chapter 3, namely $BnB(Cost)$ and $BnB(SCS)$ for random instances with different average complexities. Recall that $BnB(SCS)$ branches on an arc in the smallest cycle with the lowest upper tolerance value, and that $BnB(Cost)$ branches on an arc in the smallest cycle with the highest cost value. The complexity of the instances is artificially varied using the theory of *complexity transitions*.

Complexity transitions analysis is a statistical analysis of the complexity of $\mathcal{NP}$-hard problems (Hogg, 1996). Complexity transitions are special cases of *phase transitions*. According to the definition from Wilson (1979), a phase transition is a dramatic change of some system property; its 'state' crosses a certain virtual boundary when certain control parameters change. Phase transitions are usually studied in physical systems. An example is the melting of a solid substance (Hogg *et al.*, 1996), for which the parameter under control is the environment temperature and the output is the state of the substance. A small increase in the environment temperature across the melting temperature leads to an abrupt softening of the substance. This is a phase transition, since the transition takes place abruptly and not smoothly. A phase transition is schematically depicted in Figure 4.1. The state of the system in this figure depends on the values of two parameters. The curved line marks the border between the two states 1 and 2, for example, the state in which the substance is solid and substance is fluid.

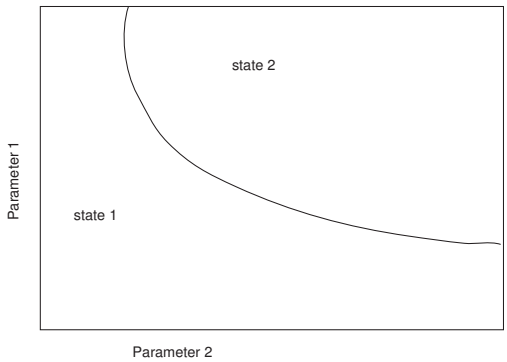

Figure 4.1. Phase transition

The study of *complexity transitions* considers a search procedure for finding

a solution of an $\mathcal{NP}$-hard problem instance as a system (Hogg *et al.*, 1996). An example of such a search procedure is Branch and Bound (Zhang and Pemberton, 1994). The input of the 'system' consists of the instance to be solved; the output is the number of nodes in the search tree visited before the instance is solved. It is assumed that the parameter choice of the random distribution from which the instances are drawn determines the complexity of the instances. Complexity theory results show that there is a thin (virtual) border between instances with search trees that are, on average, polynomial in the input size, the easy instances, and instances with exponential search trees, the hard instances. For an explanation of the terms 'polynomial' and 'exponential', we refer to Section 1.3. The purpose of the analysis is to find combinations of parameter values which mark the border between easy and hard instances. The 'abrupt' transition from easy to hard instances is called a *complexity transition*. It appears for many COPs; see, for example, Dunne *et al.* (2000) and Van der Veen (1992).

The importance of this type of complexity analysis is twofold. Firstly, it is helpful for instance generation in computational experiments. A possible pitfall in a computational experiment is that algorithms are only tested on instances with average polynomial complexity, which may lead to results that are invalid for more difficult problem instances (Johnson, 1991). Complexity analysis can be used to generate both easy and hard random instances. Secondly, since the border between easy and difficult instances is typically thin, instances situated on the exponential side of the transition may be transferred to the polynomial side by means of minor changes in the data; see Zhang and Pemberton (1994). The so-called Data Correcting Algorithms from Goldengorin (2002) work along a similar principle.

In this section, we investigate complexity transitions for the Asymmetric Traveling Salesman Problem (ATSP). Previous studies on complexity transitions for the ATSP are Zhang and Korf (1996), Zhang (2003), and Zhang (2004). In these studies, random instances drawn from both uniform and lognormal distributions are considered; we only take the uniform distribution into account here. The uniform distribution is supported on $\{1, \ldots, R\}$, where the parameter under control is R, the *range* of the distribution. Theoretical and computational results in the papers mentioned above show that search tree sizes undergo a complexity transition as the value of R changes.

We perform computational experiments on a cost-based BnB algorithm as well as on a tolerance based algorithm: $BnB(SCS)$ and $BnB(Cost)$ from Chapter 3, respectively. All instances are of size 100, and the values of c_{ij} are drawn from a uniform distribution supported on $\{1, \ldots, R\}$. The value of R runs from 10 to 10,000 in varying step sizes. One hundred instances are generated and solved for each value of R. The results are presented in Table 4.1. We compare both the average search trees and the average solution times for each algorithm and for each selected value of R. (Zhang, 1993) have already established that complexity transactions exist for the ATSP when R is the input parameter. The question is: do cost-based and tolerance-based BnB algorithms behave similarly for instances on both sides and in the complexity transition.

Table 4.1. Average search trees and solution times of $BnB(SCS)$ and $BnB(Cost)$

Range R	$BnB(SCS)$		$BnB(Cost)$	
	Search tree	Solution time	Search tree	Solution time
10	1.40	0.001	3.34	0.001
25	6.94	0.010	10.64	0.004
50	16.04	0.033	114.04	0.028
100	27.66	0.053	391.90	0.075
200	34.70	0.077	638.74	0.124
300	31.27	0.070	810.22	0.158
400	32.62	0.071	861.42	0.162
500	31.04	0.078	813.04	0.157
700	36.90	0.085	884.94	0.185
1000	34.22	0.074	1120.02	0.219
2000	39.10	0.101	1036.68	0.209
5000	37.76	0.100	1090.22	0.220
10000	34.58	0.125	995.98	0.209

Figure 4.2 presents the search tree sizes of $BnB(Cost)$ and $BnB(SCS)$ graphically. In order to make the patterns of both algorithms comparable, we set the search trees for $R = 100$ equal to 100% and compute the remaining search tree sizes relative to $R = 100$. So for $BnB(Cost)$, the relative search tree of $R = 1000$ is $\frac{1120.02}{391.90} \times 100\% = 28.58\%$.

Figure 4.2 also shows that the search tree sizes increase strongly between $R = 25$ and $R = 200$. For $R = 10$ and 25, the BnB systems are in the 'easy'

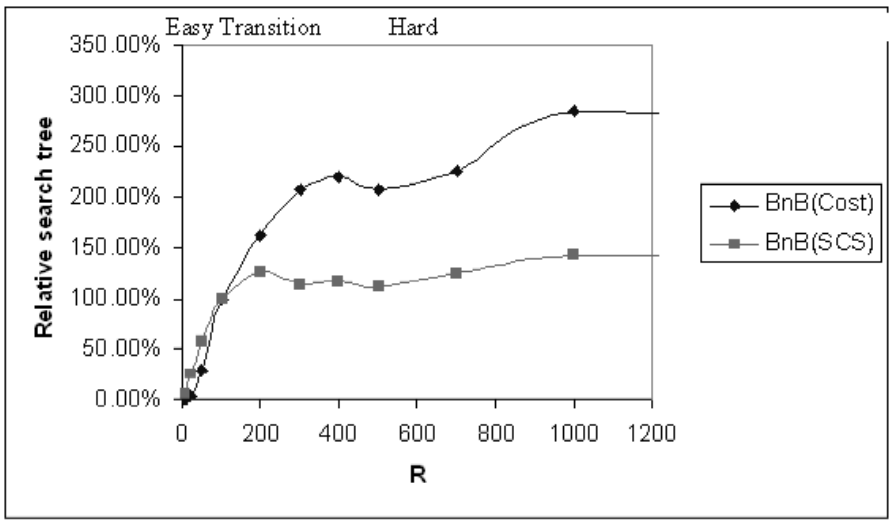

Figure 4.2. Relative search trees for $n = 100$

.

state: instances are solved quickly. When R reaches 200, the search trees and solution times are relatively large, but when R is increased beyond that, only small changes can be identified. This is the 'hard' state. The region between $R = 25$ and $R = 200$ is the *complexity transition* region. From Figure 4.2, we expect a sudden increase in search tree sizes, but the borderline is not so sharp. However, the pattern is similar to the one found in Zhang (1993).

The increase for $BnB(Cost)$ is much stronger than the increase for $BnB(SCS)$. For small values of R, the search trees of $BnB(Cost)$ and $BnB(SCS)$ are almost equally large. It appears that for small values of R, branching on tolerances does not make too much sense. Most tolerance values are equal to 0, and branching on tolerances is now nothing more than random branching. In contrast, when the value of R is large, the trees of $BnB(SCS)$ are clearly smaller than those of $BnB(Cost)$. So now it makes much more sense to branch on tolerances. Our explanation is that there are many different intercity distances, so that low cost arcs are often not part of an optimal ATSP solution. Section 3.5 shows, on the other hand, that arcs with high upper tolerances frequently belong optimal ATSP solutions.

To summarize the results of this section, it appears that the tolerance-based

BnB algorithm $BnB(SCS)$ is more powerful on relatively difficult randomly generated ATSP instances than the cost-based $BnB(Cost)$.

4.3 Hybrid Branch and Bound Algorithms

One of the main disadvantages of using the relaxation tolerances in an algorithm is the time required to compute the tolerance values. In this section, we try to restrict the tolerance computations to specific nodes of the search tree. To that end, we introduce *hybrid BnB algorithms*, which apply tolerance-based branching and bounding only at nodes where it is likely to be effective and cost-based branching and bounding at other nodes.

Which nodes in the search tree are good candidates for branching on tolerances? We have seen in Section 3.5 that, at the top nodes of the search tree, it is important to include or exclude the correct arcs. If an arc from all optimal ATSP tours is excluded in the top node, the algorithm searches through a large part of the solution space where no good ATSP solution can be found. If, on the other hand, a wrong arc is included or excluded in a bottom node of the search tree, then the consequence of this mistake is that the BnB only searches through a few additional nodes in the search tree in vain. Following this line of reasoning, it appears to be so that tolerance-based branching is effective at top nodes of the search tree, and at nodes close to the top node. Note that when upper tolerances are available for branching, they are automatically available for bounding; see Section 3.6.

We define the *depth* of a node in the search tree as the number of edges on the shortest path from that node to the top node in the search tree; see Section 2.2. We hypothesize that tolerance-based branching and bounding is most effective at small depths in the search tree, i.e., close to the top node. Let δ be a user-defined parameter. We introduce the set of hybrid BnB algorithms $HBnB(\delta)$ branch on upper tolerances if the depth of the current node is smaller than a user-defined parameter δ; branching and bounding based on costs is performed at the other nodes. If $\delta = 0$, then the algorithm is the fully cost-based $BnB(Cost)$, and if $\delta \geq n^2$, we obtain the fully tolerance-based $BnB(SCS)$. A graphical example of a hybrid BnB algorithm ($\delta = 2$) is depicted in Figure 4.3.

Another potential advantage of the choice of an intermediate value of δ may

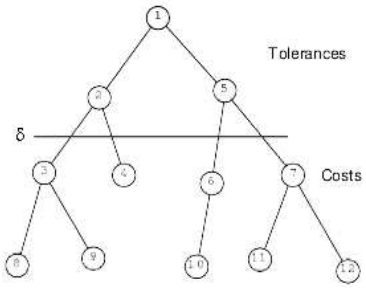

Figure 4.3. Hybrid BnB search tree ($\delta = 2$)

be the following. Since he SCS branching rule breaks a smallest cycle, the time needed to determine the SCS arc is proportional to the cardinality of the smallest cycle; see Section 3.6. When the BnB algorithm proceeds deeper into the search tree, the remaining cycles tend to be larger. Hence, the complexity of computing the SCS arc deep in the search tree is likely to be relatively high.

Computational experiments are conducted on some practical instances from the ATSPLIB (Reinelt, 1995). The value of δ is varied. For each value of δ and for each instance, search tree sizes and solution times are reported in the figures below. The tables containing the results, Table 4.4 and Table 4.3, can be found in the Appendix.

Figure 4.3 present the search tree sizes of the selected ATSPLIB instances. If applying upper tolerance-based BnB is indeed most effective at top nodes, a large decrease in the search tree size can be expected between $\delta = 0$ and $\delta = 2$ and $\delta = 5$. However, it turns out that for more than half of the tested instances, the search trees *increase* in size, when δ increases from 0 to 2. The same holds when δ is increased from 2 to 5.

Figure 4.3 present the solution times. The results indicate that the smallest solution times are often achieved by either the fully cost-based or the fully

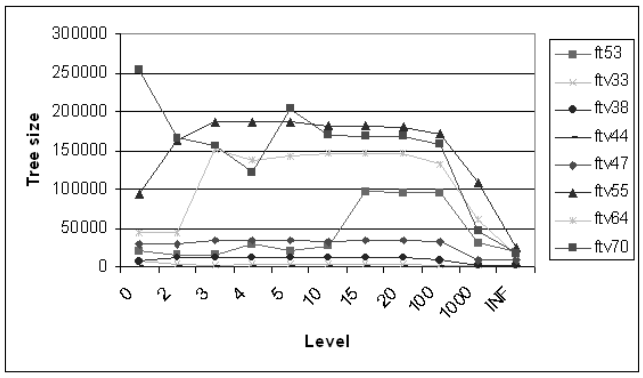

Figure 4.4. Search tree sizes of ATSPLIB instances

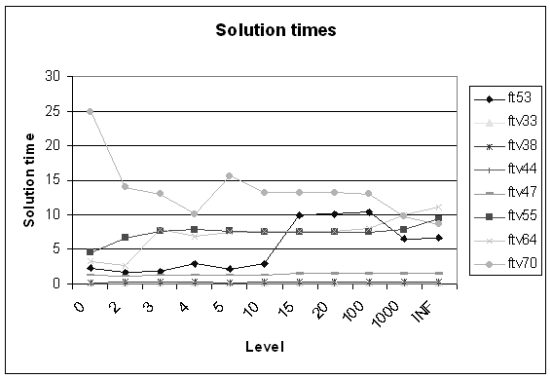

Figure 4.5. Solution times of ATSPLIB instances

tolerance-based algorithm, depending on the instance at hand. The experiments also indicate that the longest solution times are often achieved by either $BnB(SCS)$ ($\delta = \infty$) or $BnB(Cost)$ ($\delta = 0$) as well. In particular when we take $\delta = 2$, solution times are reasonable if either $BnB(Cost)$ or $BnB(SCS)$ requires relatively long solution times.

The computational experiments indicate that the largest reductions are usually not achieved for small values of δ. This finding implies that branching and bounding on upper tolerances is not only effective at top nodes of the BnB search tree, but that it should be applied at each node in the search tree. However, more extensive research is needed to provide conclusive answers. For example, it is interesting to consider randomly generated instances as well.

An interesting direction of future research is to perform *reduced tolerance*

computations, meaning that tolerance computations are performed only partially. In Chapter 3, we have already observed that there is a large degree of similarity between tolerance-based BnB algorithms for COPs and strong branching for Integer Linear Programming (ILP) problems. The *strong branching approach*, also called *pseudo-cost based approach* in Linderoth and Savelsbergh (1997), solves ILP problems as follows; see Achterberg *et al.* (2004). Assume that we use BnB to solve an ILP problem, with the usual LP relaxation. For any node k in the search tree, let X_k be the set of integer variables with fractional values, and let $x \in X_k$. If the value of a variable $x = f$, then the LP problem P_1 with the additional constraint $x \geq \lceil f \rceil$ and the problem P_2 with the constraint $x \leq \lfloor f \rfloor$ are solved. The *pseudo-cost* of the variable x is a weighted average of the increase in objective function of optimal solutions of P_1 and P_2, the weights being $\frac{1}{1-(f-\lfloor f \rfloor)}$ and $\frac{1}{f-\lfloor f \rfloor}$, respectively. The variable with the largest pseudo-cost is selected as the next branching variable. Both pseudo-cost based and tolerance based branching rules explore the objective value of the solution obtained after using each branching variable before the actual branching is performed.

Linderoth and Savelsbergh (1997) report that the use of pseudo-costs in branching reduces search trees substantially. However, the pseudo-cost computations take too much time, even though search tree reductions are large. In Linderoth and Savelsbergh (1997), it is found that reduced pseudo-cost computations lead to the smallest solution times. In reduced pseudo-cost computations, the LP-problems P_1 and P_2 are not fully solved, but only to a prespecified number of iterations of the LP-solver. A similar strategy can be followed with tolerances.

4.4 An Additive Tolerance-Based Lower Bound for the DCMSTP

In Section 3.7, upper tolerance-based lower bounds are introduced for the Asymmetric Traveling Salesman Problem (ATSP), and its Assignment Problem (AP) relaxation. These lower bounds use the concept of *offenders*, sources of infeasibility in the current relaxation solution, to construct a lower bound. The minimum cost of removing one offender is added to the lower bound. Can we also take the cost of removing more than one offender at once and still guarantee a

lower bound?

An AP solution is infeasible for the ATSP if it consists of more than one subcycle, so subcycles are offenders. The AP solution value itself is a lower bound for the ATSP, but in order to obtain an ATSP solution, at least one subcycle should be "broken". It is shown in Section 3.7, that the cost of breaking a subcycle is at least the minimum value across all upper tolerance values of arcs in that subcycle, called the *bottleneck tolerance*. This value is then added to the the optimal AP solution value in order to obtain a lower bound for optimal ATSP solution of that instance. For randomly generated instances, these tolerance-based lower bounds appear to be a good approximation of the value of optimal ATSP solutions, but for most practical instances, the gap between the lower bound and the value of the optimal solution is still wide.

When there are several subcycles in the AP solution, the lower bounds would be much tighter when the sum of the bottleneck tolerances of a set of subcycles is added simultaneously to the value of the AP solution. Unfortunately, we cannot guarantee that the value obtained is a lower bound for an ATSP solution. Consider the counterexample depicted in Figure 4.6. The number next to an arc denotes the cost of the arc. Arcs with cost 0 form the (unique) AP solution of this example, consisting of four subcycles. Consider the subcycle $(\{(1,2),(2,1)\}$. The bottleneck tolerance value is 4, achieved on the arc $(2,1)$. The cheapest AP solution after the deletion of the arc $(2,1)$ consists of an ATSP tour through all locations with cost 4; this is the unique shortest ATSP tour. The bottleneck tolerance values of the other cycles are 4 as well. So if the bottleneck tolerance of another subcycle is added as well, say $\{(3,4),(4,5),(5,3)\}$, then the value of the suggested lower bound is $0+4+4=8$, whereas the value of the optimal AP solution is 4. Even though there are four cycles, the bottleneck tolerance of only one subcycle can be added to the lower bound.

In case of the AP, it is generally not possible to add the upper tolerances from more than one offender at once and still guarantee a lower bound. Fortunately, this negative result for the AP does not extend to all COPs. We show in this section that the sum of the upper tolerances of more than one element can be added to the lower bound for problems which have the *Minimum Spanning Tree Problem* (MSTP) as a relaxation.

Given an undirected graph G with edge set $\mathcal{E}$ and vertex set V, the MSTP is

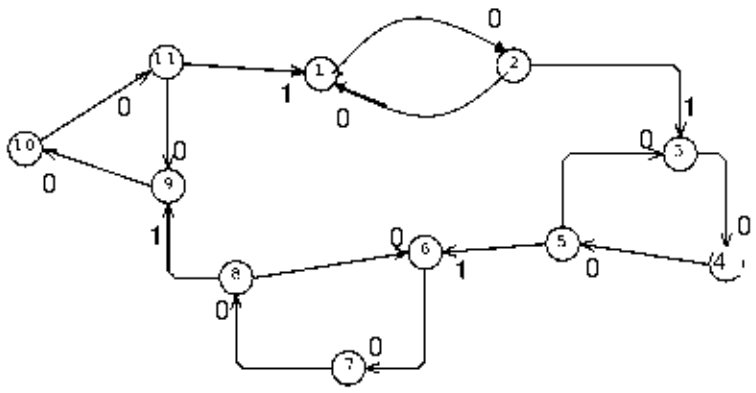

Figure 4.6. Counterexample: Removal of the arc $(2, 1)$ leads to an ATSP tour

the problem of connecting n vertices in V by means of $n-1$ edges from $\mathcal{E}$ against minimum cost. The resulting solution is called a *Minimum Spanning Tree (MST)*. The MSTP can be solved in $O(n^2)$ time with, among others, Prim's algorithm (Prim, 1957). The MSTP may serve as a relaxation of the Symmetric Traveling Salesman Problem (STSP), and of the *Degree-Constrained Minimum Spanning Tree Problem (DCMSTP)*; see for example Volgenant (1989). Let $d \geq 2$. The d-DCMSTP is the problem of finding an MST with at most d edges incident to each vertex. Applications of the d-DCMSTP are found mainly in communication networks; see Krishnamoorthy *et al.* (2001) and Caccetta and Hill (2001). The d-DCMSTP is $\mathcal{NP}$-hard for any value of $d \geq 2$ (Garey and Johnson, 1979).

In the framework of Section 3.3, the MSTP can be denoted by $(\mathcal{E}, \mathcal{T}, C, f_C)$. The set $\mathcal{E}$ denotes the set of edges, the set $\mathcal{T}$ denotes the set of spanning trees. Let n be the number of vertices to be connected. Each $T \in \mathcal{T}$ has the following well-known properties: it contains exactly $n - 1$ edges, contains no subcycles, and is connected. C refers to the *instance* of the MSTP, the cost of an edge $e \in \mathcal{E}$ is denoted by $c(e)$, and f_C denotes the cost function, i.e., $f_C(T)$ is the sum of the costs of all edges in $T \in \mathcal{T}$. We denote the set of MSTs by $\mathcal{T}^*$.

Let $E \subseteq \mathcal{E}$. Define $\mathcal{T}_{-}(E) = \{T \in T : T \cap E = \emptyset\}$, and let $\mathcal{T}^*_{-}(E)$ be the set of MSTs in $\mathcal{T}_{-}(E)$. Clearly, $\mathcal{T}_{-}(\emptyset) = \mathcal{T}$ and $\mathcal{T}^*_{-}(\emptyset) = \mathcal{T}^*$. When we consider the

set of edges $\{e_1, \ldots, e_k\}$, we write $\mathcal{T}_-(e_1, \ldots, e_k)$ and $\mathcal{T}^*_-(e, \ldots, e_k)$ instead of $\mathcal{T}_-(\{e_1, \ldots, e_k\})$ and $\mathcal{T}^*_-(\{e_1, \ldots, e_k\})$, respectively. Take any $T^* \in \mathcal{T}^*$. Since the MSTP is a monotonous COP (see Section 3.3), the upper tolerance $u_{T^*}(e)$ of any arc $e \in T^*$ can be computed as follows (see e.g. Goldengorin *et al.* (2006)):

$$u_{T^*}(e) = f_C[T^*_-(e)] - f_C[T^*]. \tag{4.4.1}$$

The following lemmas include convenient characteristics of upper tolerances of MSTP solutions.

Lemma 4.4.1. *Let $T \in \mathcal{T}^*$ and let $e \in T$. Moreover, let $\mathcal{T}_-(e) \neq \emptyset$. There is $T_- \in \mathcal{T}^*_-(e)$ such that $|T \oplus T_-| = 2$.*[1]

Lemma 4.4.1 is proven in, among others, Helsgaun (2000). Let $T_1, T_2 \in \mathcal{T}$, $e \in T_1$ and $\mathcal{T}_-(e) \neq \emptyset$. T_2 is called a *maximal intersection* tree of T_1 w.r.t. e. We define $MIT(T_2, e)$ as follows: $T_2 \in MIT(T_2, e)$, iff $|T_1 \oplus T_2| = 2$. Moreover, for each edge e, we denote and define the substitute edge of e as $p_{T_1}(e) \in T_2 \backslash T_1$. Clearly, $T_1 \oplus T_2 = \{e, p_{T_1}(e)\}$.

When $T^*_-(e)$ is a maximal intersection tree of T^*, it holds that $T^* \oplus T^*_-(e) = \{e, p_{T^*}(e)\}$. It follows from (4.4.1) that $u_{T^*}(e) = f_C[T^*_-(e)] - f_C[T^*] = f_C[T^*] + c[p_{T^*}(e)] - c(e) - f_C[T^*] = c[p_{T^*}(e)] - c(e)$.

Lemma 4.4.2. *Let $e \in T^* \in \mathcal{T}^*$, and let $T^*_-(e) \in \mathcal{T}^*_-(e) \cap MIT(T^*, e)$. Let $a \in T^* \cap T^*_-(e)$ with $\mathcal{T}_-(a, e) \neq \emptyset$. It then holds that $u_{T^*_-(e)}(a) \geq u_{T^*}(a)$.*

Proof. By definition, $p_{T^*}(a) = T^*_-(a) \setminus T^*$ for any maximal intersection tree $T^*_-(a)$ w.r.t. $a \in T^*$. To prove the lemma, we use Lemma 4.4.1 and distinguish between the following two cases.

Case 1: $p_{T^*}(a) \neq p_{T^*}(e)$. Then $u_{T^*_-(e)}(a) = f_C[T^*_-(a, e)] - f_C[T^*_-(e)] = c[p_{T^*}(a)] - c[a] = (f_C[T^*] + c[p_{T^*}(a)] - c[a]) - f_C[T^*] = f_C[T^*_-(a)] - f_C[T^*] = u_{T^*}(a)$. Hence, $u_{T^*_-(e)}(a) = u_{T^*}(a)$.

Case 2: $p_{T^*}(a) = p_{T^*}(e)$. From Lemma 4.4.1, it follows that $\exists T^*_-(e) \in \mathcal{T}^*_-(e)$, such that $T^*_-(e) := T^* \backslash \{e\} \cup \{p_{T^*}(e)\}$. Since $p_{T^*}(a) = p_{T^*}(e), p_{T^*}(a) \in T^*_-(e)$. As a consequence, $T^*_-(e) \cup \{p_{T^*}(a)\} \setminus \{a\} \notin \mathcal{T}$, since it contains

[1] $\oplus$ denotes the symmetric difference, $A \oplus B = (A \cup B) \setminus (A \cap B)$.

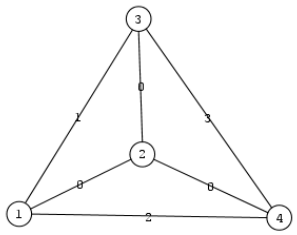

Figure 4.7. MSTP example

only $n-2$ edges. So a obtains a new substitute edge $p_{T^*_-(e)}(a) \neq p_{T^*}(a)$ It holds that $p_{T^*_-(e)}(a) = T^*_-(a,e) \setminus T^*_-(e)$ for the maximal intersection tree $T^*_-(a,e) \in \mathcal{T}^*_-(a,e)$ of $T^*_-(a)$ w.r.t. a. We now prove by contradiction that $c[p_{T^*_-(e)}(a)] \geq c[p_{T^*}(a)]$.

Assume that $c[p_{T^*_-(e)}(a)] < c[p_{T^*}(a)]$. Let $T^*_-(a) \in \mathcal{T}^*_-(a)$. Let $T^*_- := T^* \setminus \{a\} \cup \{p_{T^*_-(e)}(a)\}$. Clearly, $a \notin T^*_-$. Then, $f_C(T_-) = f_C(T^*) + c[p_{T^*_-(e)}(a)] - c[(a)] < f_C(T^*) + c[p_{T^*}(a)] - c[(a)] = f_C[T^*_-(a)]$. This is a contradiction to the condition that $T^*_-(a) \in \mathcal{T}^*_-(a)$. Therefore, $c[p_{T^*_-(e)}(a)] \geq c[p_{T^*}(a)]$, and $u_{T^*_-(e)}(a) = f_C[T^*_-(a,e)] - f_C[T^*_-(e)] = c[p_{T^*_-(e)}(a)] - c[a] \geq c[p_{T^*}(a)] - c(a) = u_{T^*}(a)$.

In both cases, $u_{T^*_-(e)}(a) \geq u_{T^*}(a)$. □

The assumption that $T^*_-(e)$ is a maximum intersection tree is necessary, because there may exist $T^*_-(e) \in \mathcal{T}^*_-(e)$ and $a \in T^* \setminus \{e\}$, such that $a \notin T^*_-(e)$. By definition, the upper tolerance of such an edge a satisfies $u_T(a) = \infty$, whereas $u_{T^*}(a) = 0$.

Lemma 4.4.2 implies that, if $p_{T^*}(e_1) \neq p_{T^*}(e_2)$, then $u_{T^*_-(e_1)}(e_2) = u_{T^*}(e_2)$. This property is useful in tolerance-based BnB algorithms, because many upper tolerance values can be stored during branching steps instead of being recomputed. This storage of upper tolerance values may save large amounts of computing time.

We illustrate Lemma 4.4.2 with the following small example, depicted in Figure 4.7. Clearly, $T^* = \{(1,2),(2,3),(2,4)\}$ with $f_C(T^*) = 0$ is a unique MST. Moreover, $u_{T^*}(1,2) = 1$, $u_{T^*}(2,3) = 1$, $u_{T^*}(2,4) = 2$; $p_{T^*}(1,2) =$

$p_{T^*}(2,3) = (1,3)$, and $p_{T^*}(2,4) = (1,4)$. In the example, $(1,2)$ and $(2,3)$ have the same substitute edge $(1,3)$. When $(1,2)$ is forbidden, the substitute edge becomes $p_{T^*_-(1,2)}(2,3) = (3,4)$. As a consequence, the upper tolerance value of $(2,3)$ increases from 1 to 3.

Lemma 4.4.3. *Take any $e \in T^* \in \mathcal{T}^*$. Then for each $a \in T^* \setminus \{e\}$ such that $u_{T^*}(a) > 0$, it holds that $a \in \cap\mathcal{T}^*_-(e)$.*

Proof. If $\mathcal{T}_-(a,e) = \emptyset$, then $u_{T^*_-(e)}(a) = \infty$, meaning that $a \in \cap\mathcal{T}^*_-(e)$. So assume that $\mathcal{T}_-(a,e) \neq \emptyset$.

Assume to the contrary that there is a tree $T \in \mathcal{T}^*_-(e)$ such that $a \notin T$. Since $T \in \mathcal{T}^*_-(a)$ and $e \notin T$, it follows that $T \in \mathcal{T}_-(a,e)$.

Take any $T^*_-(a,e) \in \mathcal{T}^*_-(a,e)$ and any $T^*_-(a) \in \mathcal{T}^*_-(a)$. Lemma 4.4.2 implies that $u_{T^*_-(e)}(a) \geq u_{T^*}(a)$. Clearly, $f_C[T] \geq f_C[T^*_-(a,e)]$, since $T \in \mathcal{T}_-(a,e)$. So we have that $f_C[T] \geq f_C[T^*_-(a,e)] = f_C[T^*_-(e)] + u_{T^*_-(e)}(a) \geq f_C[T^*_-(e)] + u_{T^*}(a) > f_C[T^*_-(e)]$, which contradicts the condition that $T \in \mathcal{T}^*_-(e)$. □

Let $E = \{e_1, \ldots, e_k\} \subseteq T^*$. We denote and define the *upper tolerance of a set of edges* E as $u_{T^*}(E) = f_C[T^*_-(E)] - f_C[T^*]$. Given two sets $E_1, E_2 \subseteq T^*$, we have that, if $E_1 \subseteq E_2$, then $\mathcal{T}_-(E_2) \subseteq \mathcal{T}_-(E_1)$, therefore, $u_{T^*}(E_1) \leq u_{T^*}(E_2)$.

Lemma 4.4.4. *Let T^* be an MST and assume that $T^*_-(E) \neq \emptyset$ for some $E \subset \mathcal{E}$. Let $E = \{e_1, \ldots, e_k\}$. Then $u_{T^*}(E) = u_{T^*}(e_1) + u_{T^*_-(e_1)}(e_2) + \ldots + u_{T^*_-(e_1,\ldots,e_{k-1})}(e_k)$.*

Proof. Since $T^*_-(E) \neq \emptyset$, it follows for each $\bar{E} \subseteq E$ that $\mathcal{T}_-(\bar{E}) \neq \emptyset$. So all upper tolerance values $u_{T^*}(\bar{E})$ are finite.

Note that for each $j = 1, \ldots, k-1$, it holds that $u_{T^*_-(e_1,\ldots,e_j)}(e_{j+1}) = f_C[T^*_-(e_1, \ldots, e_{j+1})] - f_C[T^*_-(e_1, \ldots, e_j]$. Then,

$$\begin{aligned} u_{T^*_-(e_1,\ldots,e_k)} &= f_C[T^*_-(e_1,\ldots,e_k)] - f_C(T^*) \\ &= (f_C[T^*_-(e_1,\ldots,e_k)] - f_C[T^*_-(e_1,\ldots,e_{k-1})]) + \\ &(f_C[T^*_-(e_1,\ldots,e_{k-1})] - f_C[T^*_-(e_1,\ldots,e_{k-2})]) + \ldots \\ &\quad -f_C[T^*_-(e_1)] + (f_C[T^*_-(e_1)] - f_C(T^*)) \\ &= u_{T^*}(e_1) + u_{T^*_-(e_1)}(e_2) + \ldots + u_{T^*_-(e_1,\ldots,e_{k-1})}(e_k). \end{aligned} \tag{4.4.2}$$

□

Lemma 4.4.4 guarantees that for each subset $E \subset T^*$ and each $e \in T^* \setminus E$, $0 < u_{T^*_-(E)}(e) < \infty$, under the assumption that $\mathcal{T}_-(E \cup \{e\}) \neq \emptyset$.

Based on the four previous lemmas, we are able to construct new lower bounds for the STSP and the d-DCMSTP. Assume that the degree of a vertex $v \in V$ exceeds d. For instance, the maximum degree is 3, whereas the actual degree of v is 7. In order to 'reduce' the degree of v to 3, at least four edges incident to it have to be removed. According to Theorem 4.4.1, the minimum cost of removing four edges is at least equal to the sum of the smallest four upper tolerance values of edges incident to v.

More formally, let $v \in V$ be an arbitrary vertex in G and let $d_{T^*}(v)$ be the degree of v in T^*. Define $I_{T^*}(v) \subseteq T^*$ as the set of edges in the tree T^* incident to v. Order the edges in $I_{T^*}(v)$ in a non-decreasing order of upper tolerance values and obtain $u_{T^*}(e^1) \leq \ldots \leq u_{T^*}(e^{d_{T^*}(v)})$. Theorem 4.4.1 adds the lowest $d_{T^*}(v) - d$ upper tolerance values to $f_C[T^*]$ to obtain a lower bound of the d-DCMSTP solution value $f_C(D^*_d)$.

Theorem 4.4.1. *Let T^* be an optimal solution of the MSTP instance C, and let D^*_d be an optimal solution of the d-DCMSTP instance with the same C. Let $v \in V$ such that $d_{T^*}(v) > d$. Denote by $I^+_{T^*}(v) = \{e \in I_{T^*}(v) : \ u_{T^*}(e) > 0\}$ the set of edges incident to v with positive upper tolerance values. Let $k = |I^+_{T^*}(v)|$, and let $e^1, \ldots, e^k \in I^+_{T^*}(v)$ such that $0 < u_{T^*}(e^1) \leq \ldots \leq u_{T^*}(e^k)$. It then holds that $f_C(T^*) + \sum_{i=1}^{k-d} u_{T^*}(e^i) \leq f_C(D^*_d)$.*

Proof. Take any vertex v such that $d_{T^*}(v) > d$. By definition, $I_{T^*}(v) \cap D^*_d \leq d$, so at least $d_{T^*}(v) - d$ edges incident to v should be deleted in order to obtain a d-DCMSTP solution D^*_d. This implies that at least $k - d$ edges from $I^+_{T^*}(v)$ must be deleted.

There is a subset $E = \{e_1, \ldots, e_{k-d}\} \subset I^+_{T^*}(v)$ such that $E \cap D^*_d = \emptyset$. Clearly, $D^*_d \in \mathcal{T}_-(E)$, so that for each $T^*_-(E) \in \mathcal{T}^*_-(E)$, we have that $f_C[T^*_-(E)] \leq f_C[D^*_d]$.

Assume, without loss of generality, that $E \cap D^*_d = \emptyset$. It then holds that

$$f_C[T^*_-(E)] = f_C(T^*) + u_{T^*}(E) \leq f_C(D^*_d).$$

Since $D^d \in \mathcal{T}^*_-(E)$, it holds that $f_C[T^*_-(E)] \leq f_C(D^*_d)$. The term $u_{T^*}(E)$ can be bounded as follows:

$$u_{T^*}(E) = u_{T^*}(e_1, \ldots, e_{k-d}) =$$
$$u_{T^*}(e_1) + u_{T^*_-(e_1)}(e_2) + \ldots + u_{T^*_-(e_1,\ldots,e_{k-d-1})}(e_{k-d}) \quad (4.4.3)$$
$$\geq u_{T^*}(e_1) + \ldots + u_{T^*}(e_{k-d}) \quad (4.4.4)$$
$$\geq \sum_{i=1}^{k-d} u_{T^*}(e^i) \quad (4.4.5)$$

By Lemma 4.4.4, the value of $u_{T^*}(E)$ is independent of the order of deleting edges from E. It follows from Lemma 4.4.3 that $u_{T^*_-(e_1,\ldots,e_{i-1})}(e_i)$ is well defined for each $i = 2, \ldots, k-d$. Moreover, Lemma 4.4.2 states that $u_{T^*_-(e_1,\ldots,e_{i-1})}(e_i) \geq u_{T^*}(e_i)$ for each $i = 1, \ldots, k-d$. It follows from Lemma 4.4.2 that the term in (4.4.4) is at most equal to the term in 4.4.3. Finally, (4.4.5) holds, because we select the $k-d$ edges in $I^+_{T^*}(v)$ with the smallest upper tolerance values.

Since $u_{T^*}(E) \geq \sum_{i=1}^{k-d} u(e^i)$, it holds that $f_C(D^*_d) \geq f_C[T^*_-(E)] = f_C(T^*) + u_{T^*}(E) \geq f_C(T^*) + \sum_{i=1}^{k-d} u(e^i)$. □

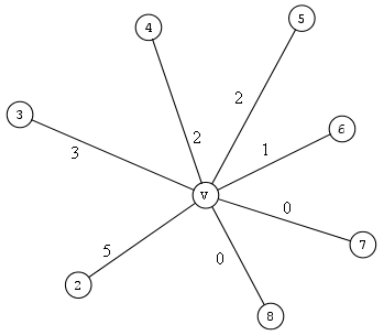

Figure 4.8. Vertex with seven incident edges in T^*

We illustrate Theorem 4.4.1 with the following example. In T^*, seven edges are adjacent to v, and the upper tolerance values are, in non-increasing order, $5, 3, 2, 2, 1, 0, 0$; see Figure 4.4. A DCMSTP solution with $d = 3$ is to be obtained. There are five edges in $I^+_{T^*}(v)$, so the sum of two smallest nonzero upper

tolerance values, $2+1$, is added to the value of the MSTP solution such as to form a lower bound for any 3-DCMSTP solution value $f_C(D_3^*)$.

In Theorem 4.4.1, we take the sum of the upper tolerances of $k-d$ edges $e \in I^+_{T^*}(v)$ instead of $d_{T^*}(v)-d$ edges from $I_{T^*}(v)$. The lower bounds obtained with both approaches are clearly the same. However, when only nonzero upper tolerances are considered, Theorem 4.4.1 is much easier to prove.

Theorem 4.4.1 can be applied in a similar way to construct a lower bound for both the STSP and the d-DCMSTP. The lower bound for the STSP is obtained by taking a vertex $v \in V$ with $d_{T^*}(v) > 2$, and by adding the $d_{T^*}(v) - 2$ smallest upper tolerance values of edges in $I_{T^*}(v)$. Note that only upper tolerances with respect to T^* need to be computed.

We conjecture that the lower bound can be improved further by not merely taking a single vertex into account, but each vertex with degree larger than d, i.e., all offenders. Let $T^* \in \mathcal{T}^*$, and $d \geq 2$. Let D_d^* be an optimal solution of the corresponding d-DCMSTP instance. Define $V^+ = \{v \in V : d_{T^*}(v) > d\}$. Denote the vertices in V^+ by $v_1, \ldots, v_{|V^+|}$. For each $v \in V^+$ and $S(v) \subseteq I_{T^*}(v)$, order the edges $e(S(v), i) \in S(v),\ i = 1, \ldots, |S(v)|$ in such a way that $u_{T^*}(e(S(v), 1)) \leq \ldots \leq u_{T^*}(e(S(v), |S(v)|))$. Define for each $v \in V^+$ the function $\Delta : T^* \to \mathbb{R}$ by

$$\Delta[S(v)] = \sum_{i=1}^{|S(v)|-d} u_{T^*}(e(S(v), i)) \tag{4.4.6}$$

We require $|S(v)| > d$, otherwise $\Delta[S(v)] = 0$. For each $v_k \in V^+$, let $W(v_k) := \{(v_i, v_k) \in I_{T^*}(v_k) :\ v_i \in V^+,\ i = 1, \ldots, k-1\}$.

Conjecture

$$f_C[T^*] + \sum_{v \in V^+} \Delta[I_{T^*}(v) \setminus W(v)] \leq f_C[D_d^*] \tag{4.4.7}$$

Clearly, when $|V^+| = 1$, the conjecture reduces to Theorem 4.4.1.

We illustrate the conjecture with the following example. Suppose that in T^*, $|V^+| = 2$, and $v, w \in V^+$. When $(v, w) \notin T^*$, then the conjecture states that both the $d_{T^*}(v) - d$ smallest upper tolerances incident to v and the $d_{T^*}(w) - d$ smallest upper tolerances of edges incident to w can be added to the lower bound.

A problem arises when for $v, w \in V^+$, the edge e between v and w belongs to T^*, i.e., $e = (v, w) \in T^*$. In order to prevent double-counts, (v, w) should be assigned to either the incidence set of v or that of w. Take $d = 3$, $d_{T^*}(v) = 4$, $d_{T^*}(w) = 6$, and $(v.w) \in T^*$. Suppose that the upper tolerance values of the edges incident to v are $5, 3, 3, 1$, of the edges incident to w $4, 4, 3, 2, 2, 0$, and suppose that $u_{T^*}(e) = 3$. If we assign e to v, then $W(w) = \{e\}$ and $W(v) = \emptyset$. It means that sum of the smallest upper tolerance value of the edges in $I_{T^*}(v)$ and the two $(= d_{T_*}(w) - d - 1)$ smallest upper tolerance values from edges in $I_{T^*}(w) \setminus \{e\}$, can be added to the value of $f_C[T^*]$. The value of the lower bound is then $f_C[T^*] + 1 + (2 + 0) = f_C[T^*] + 3$. If, however, we assign e to $W(w)$ ($W(w) = \emptyset$), then the lower bound is $f_C[T^*] + 0 + 2 + 2 = f_C[T^*] + 4$. The pseudo-code for constructing the conjectured lower bounds is given in Algorithm 1.

Algorithm 1 Method for constructing a lower bound for the d-DCMSTP

INPUT
  d User-specified maximum degree of each vertex;
  C Cost matrix;

MAIN ALGORITHM
  Compute MSTP solution T^*;
  Obtain for each $v \in V$ the value of $d_{T^*}(v)$; obtain $f_C(T^*)$.
  lb := $f_C(T^*)$.
  Let V^+ be the set of vertices s.t. $d_{T^*}(v) > d$;
  Make an ordering $v_1, \ldots, v_{|V^+|}$ of the vertices in V^+.
  for $i = 1$ **to** $|V^+|$
    for each $j > i$ s.t. $(v_i, v_j) \in T^*$
      assign (v_i, v_j) to v_i;
      $d_{T^*}(v_j) := d_{T^*}(v_j) - 1$;.
      $I_{T^*}(v_j) := I_{T^*}(v_j) \setminus \{(v_i, v_j)\}$; (Comment: subtract $W(v_j)$ from $I_{T^*}(v_j)$)
    end;
    for each $e \in I_{T^*}(v_i)$
      compute $u_{T^*}(e)$;
    Order upper tolerance values of edges $e \in I_{T^*}(v_i)$ in non-increasing order;
    $sumtol$:= sum of $d_{T^*}(v_i) - d$ smallest upper tolerance values;
    lb := $lb + sumtol$;
  end;
end.

OUTPUT

Lower bound lb.

Algorithm 1 depends on the order in which the vertices are considered. In the example above, the lower bound is $f_C[T^*] + 4$ when $e \in W(w)$ and $f_C[T^*] + 3$ when $e \in W(v)$.

Table 4.2. Quality of new bound for the STSP

Instance	MST	1-Tree	New bound	STSP	Gap 1-Tree STSP bridged
att48	2698	2737	7344	10628	58.38%
bays29	1557	1623	1696	2020	18.39%
berlin52	6066	6319	6432	7542	9.24%
bier127	94680	94939	95987	118282	4.49%
brazil58	9648	10128	10579	25395	2.95%
eil101	528	536	546	629	10.75%
eil51	359	369	378	426	15.79%
eil76	454	463	471	538	10.67%
fri26	741	834	841	937	6.80%
kroB100	19203	19332	19631	22141	10.64%
kroC100	18354	18642	18932	20749	13.76%
kroD100	18552	18756	19138	21294	15.05%

We have chosen a lexicographic strategy in the experiments below, i.e., the first vertex in the matrix is ranked first. For future research, other strategies should be considered, such as an ordering of the vertices in non-decreasing order of their degrees in T^*.

Table 4.2 shows the results of our conjectured bound for symmetric instances from the TSPLIB (Reinelt, 1991). The *1-Tree* is obtained as follows. One vertex, say v^*, is removed from the vertex set V and the edges incident to v^* are removed from $\mathcal{E}$. The MSTP problem is solved on the new instance with $n-1$ vertices. Finally, the two lowest cost edges incident to v^* are added to the MSTP solution and the 1-Tree is obtained. The "new bound" in Table 4.2 first constructs a 1-tree and then constructs the lower bound described in the conjecture. Except for vertex v^*, the procedure is the same as the one described in Algorithm 1. The last column reports the percentage of the gap between the values of the 1-Tree bound and the value of the optimal STSP solution bridged by the conjectured lower bound.

The results on these small instances show that only a small part of the gap between the MST solution and the STSP solution is bridged with the upper bound. For the STSP, the usual bound from Held and Karp (1970) is very tight. For the DCMSTP, however, the MST lower bound is often still used, although in Volgenant (1989), a Lagrangian relaxation is used to tighten the lower bound. When

it is proven that the conjecture is true, the application of the lower bound to the DCMSTP is a promising direction of research.

4.5 Conclusion

In this chapter, we discuss additional topics on tolerance-based algorithms.

In Section 4.2, we have conducted experiments on randomly generated ATSP instances. It is found in, among others, Zhang (1993), that there is a thin boundary between $\mathcal{NP}$-hard problem instances that are difficult by nature and instances that are easily solvable. The difficulty depends on the number of different intercity distances, which in turn depend on the range R of the uniform distribution from which the distances are drawn. The experiments indicate that an increase in R leads to far smaller increases in solution times of our tolerance-based BnB algorithm than our cost-based benchmarks.

Section 4.3 determines at which nodes in the BnB search tree the use of tolerances effective. Section 3.6 shows that predictions based on tolerances are more accurate than predictions based on costs. In Depth First Search algorithms the choice at top nodes of the search tree is crucial. For this reason, we restrict the use of tolerances to top nodes of the tree, and apply cost-based branching and bounding to other nodes. Our computational experiments indicate that for a minority of instances these hybrid BnB algorithms are more effective than the pure tolerance-based variants. However, more experiments are needed to gain better understanding of this phenomenon.

Finally, in Section 4.4, we discuss procedures for increasing the tolerance-based lower bounds from Section 3.6. Instead of adding the upper tolerance value of a single element, we propose a procedure for adding the sum of several elements to the value of the relaxation solution. Such a procedure is generally invalid for the ATSP and its AP relaxation. However, we show that the procedure can be applied successfully for Minimum Spanning Tree Problems, which are relaxations of Symmetric TSPs and Degree-Constrained Minimum Spanning Tree Problems.

Appendix

Table 4.3. Search tree sizes for various values of δ

	Level							
Instance	0	2	5	10	20	100	1000	∞
ft53	20111	15967	20671	27403	95173	95045	30831	19200
ftv33	7065	2570	3741	3570	3378	1732	1362	1362
ftv38	6195	11894	12005	11976	11602	8080	2091	2091
ftv44	619	824	793	764	368	171	171	171
ftv47	29025	29407	33342	31739	33817	31605	7692	7692
ftv55	92447	162421	185662	180543	180339	171781	108583	23034
ftv64	43441	43788	143165	145612	145202	132776	60546	15509
ftv70	253873	165683	202568	169157	167525	156989	46021	17694

Table 4.4. Solution times for various values of δ

	Level							
Instance	0	2	5	10	20	100	1000	∞
ft53	2.31	1.70	2.20	2.86	10.16	10.49	6.59	6.70
ftv33	0.22	0.05	0.11	0.11	0.11	0.11	0.11	0.16
ftv38	0.22	0.27	0.22	0.27	0.27	0.38	0.38	0.38
ftv44	0.00	0.05	0.05	0.00	0.05	0.05	0.05	0.00
ftv47	1.26	1.15	1.32	1.32	1.43	1.43	1.54	1.54
ftv55	4.51	6.70	7.64	7.47	7.47	7.53	7.75	9.51
ftv64	3.19	2.64	7.47	7.64	7.58	7.91	10.00	11.10
ftv70	24.95	13.96	15.60	13.13	13.13	13.02	9.84	8.68