Solving the Minimum Label Spanning Tree Problem by Ant Colony Optimization

A. M. Chwatal[1], **G. R. Raidl**[1]
[1]Institute of Computer Graphics and Algorithms,
Vienna University of Technology, Vienna, Austria
Email: {chwatal|raidl}ads.tuwien.ac.at

Keywords: Ant Colony Optimization, Minimum Label Spanning Tree Problem, Metaheuristics, Combinatorial Optimization

Abstract— *The Minimum Label Spanning Tree Problem is a well known combinatorial optimization problem, having applications in telecommunication network design and data compression. The problem is $\mathcal{NP}$-hard and cannot be approximated within a constant factor. In this work we present the application of ant colony optimization to this problem. Different pheromone models and construction mechanisms are introduced and local improvement methods are considered. An experimental investigation of the outlined components and a comparison to existing work are presented.*

1. Introduction

The Minimum Label Spanning Tree (MLST) problem is a well known combinatorial optimization problem, defined as follows. We are given a graph $G = (V, E)$ and a labelling function $l : E \to L$ assigning to each edge $e \in E$ an element from a discrete set L, whose elements are usually called *labels* or *colors*. The objective is to determine a spanning tree whose edges induce a least cardinality set $L^* \subset L$. The problem may also be formulated in terms of a connected subgraph instead of the spanning tree as a requirement, as any spanning tree derived from a connected subgraph spanning all vertices is equal w.r.t. the objective function.

The MLST problem was first introduced in [4] where its $\mathcal{NP}$-completeness has been shown by reduction from the set covering problem. Furthermore the non-existence of a constant-factor approximation has been shown therein. The approximation bound has been further improved in [12] and [15]. So far, many heuristic and metaheuristic methods have been applied to the MLST problem.

A constructive heuristic, called *MVCA-heuristic*, has been first introduced in [4], and further studied in [12], [15], and [8]. Its principle is to start with an empty set of labels, and then iteratively add labels and their corresponding edges that reduce the number of connected components of the graph until the graph is connected. Various genetic algorithms have been developed in [16] and [13]. Methods based on local search have been treated from a theoretical point of view in [1], and from a more practical one in [3], [9], [7], and [6]. In particular, the latter publications also cover metaheuristics like greedy randomized search procedures, local search, variable neighborhood search and the pilot method.

In [4] the authors also introduce an exact method based on the A*-algorithm. This method is very effective for instances with very few labels within the optimal solution, but impracticable for instances with many labels or higher optimal objective function values. Further exact methods are based on mathematical programming techniques: A mixed integer programming formulation based on single commodity flow has been proposed in [2], a branch-and-cut algorithm for a directed variant of the problem has been proposed in [5].

In this work, we primarily focus on *ant colony optimization*, a metaheuristic technique which has not been applied to the MLST problem so far. Algorithmic details, as well as new neighborhood structures to be used within the algorithm are presented in Section 2. Computational results are then presented in Section 3, and conclusive remarks are given in Section 4.

2. Ant Colony Optimization

Many metaheuristic techniques are based on parallel construction or modification of a pool of candidate solutions and frequently are inspired by nature. Ant colony optimization (ACO) is based on the foraging behaviour of ants. Despite the ants disability to survey the region around the anthill, or to make educated decisions, almost all ants run on near-optimal paths from the anthill to the food source and back. This "optimization task" is mainly achieved by *stigmergy*, i.e. communication by means of modification of the common environment. Along their way, each ant deposits *pheromone*, a volatile chemical factor, further ants follow trails of high pheromone concentration with high probability, but may also leave this trails to explore further ones. The latter mechanism is important to retain flexibility as well as to avoid premature convergence to a local optimum, i.e. all ants walking on a suboptimal path.

Ant Colony Optimization is a stochastic model-based search procedure, where construction steps are influenced by local information and global pheromone values. The correspondence to ants are artificial agents constructing the

solution, mostly by traversing some kind of construction graph. Various algorithmic variations with minor differences according to pheromone deposition and evaporation do exist, like $\mathcal{MAX}-\mathcal{MIN}$ Ant System, Ant Colony System, Elitist Ant System or Rank-based Ant System. For a comprehensive introduction to ACO see [11]. The algorithm proposed within this work does not directly fit into one of the mentioned categories, but contains elements from all of them. The general algorithmic template works as specified by Algorithm 1.

Algorithm 1: Generic-ACO

```
1 for it iterations do
2     for each ant m, 1 ≤ m ≤ M do
3         solution construction (of ant m)
4         optional local search
5     end
6     pheromone update
7 end
```

For our considered problem, in step "solution construction", line 3 of Algorithm 1, further labels are added iteratively, starting from an empty set of labels. The decision which label to take next is based on a probability distribution defined over all labels reducing the number of components implied by the labels (and corresponding edges) of the current partial solution, parametrized by pheromone values τ and local information η, to be defined in detail subsequently. In the following we describe the algorithm in detail, starting with the description of underlying pheromone models.

2.1 Pheromone Models

The most natural or obvious formulation is to introduce a pheromone value for each label $l \in L$, i.e. $\tau_l,\ l \in L$ (model P-I). This representation has the advantage of being very compact, but however, mutual dependencies are not represented very accurately. This might be a problem as within this simple pheromone model the algorithm cannot gather information of particular labels being of high potential importance w.r.t. some constructed subset $L' \subset L$, but being unimportant regarding other subsets $L'' \subset L$. Such information can be better reflected by the following larger pheromone model (P-II). Let τ_{ij} denote the entries of a $|L| \times |L|$ matrix. Let us again assume, we already have constructed label set L' and are about to decide which label to take next. The desire to add label l is then given by

$$\tau(L', l) := \sum_{i \in L'} \tau_{il}. \tag{1}$$

A third alternative arises, when considering the main purpose of pheromones as biasing the solution process of the MVCA-heuristic (P-III). Hence we can introduce a $|L^{\mathrm{mvca}}| \times |L|$ matrix, were L^{mvca} denotes the set of labels constructed by the MVCA-heuristic. Each row i of this matrix then provides a probability distribution for all labels to be used in the i-th construction step of each ant.

2.2 Solution Construction

In each iteration M ants construct solutions based on pheromones and local information. Let $\tau(L', l) := \tau_l$ for pheromone model (P-I), and $\tau(L', l) := \tau_{|L'|,l}$ for (P-III). Having constructed labels L' the probability for ant m of adding label l is given by

$$p^m_{L',l} = \begin{cases} p(L', l) & \text{if } c(L' \cup l) < c(L'), \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

with

$$p(L', l) = \frac{\tau(L', l)^\alpha \cdot \eta(L', l)^\beta}{\sum_{l' \in \{l \in L | c(L' \cup l) < c(L')\}} \tau(L', l')^\alpha \cdot \eta(L', l')^\beta}, \tag{3}$$

and $c(L')$ denoting the number of connected components of graph $G = (V, E')$ where E' denotes the set of edges with a label from set L'. The balance between pheromone information $\tau(L', l)$ and local information $\eta(L', l')$ is controlled by parameters α and β. Let $E(L') \subset E$ denote the subset of edges having associated to a label $l \in L'$, and let $c(L')$ denote the number of connected components of the subgraph $G' = (V, E(L'))$. Local information is then defined by expression

$$\eta(L', l) = c(L') - c(L' \cup l). \tag{4}$$

Hence, if setting $\alpha = 0$ we obtain a randomized greedy heuristic, essentially following the greedy criterion from the MVCA-heuristic. In general, parameters α and β balance between following the already gathered global information provided by pheromone trails and the local information resulting from direct improvements of the label w.r.t. the current partial solution. Using simpler models for the local information, like simply considering the number of edges associated to the particular label turned out not to work well in preliminary tests.

The most obvious way to perform the construction process is to add further labels to an initially empty set L' until a feasible solution is obtained (C-I). However, solutions having more labels than the best-so-far solution likely do not contain any useful information with regard to finding further improvements and should therefore deposit no pheromone values. Even solutions having the same number of labels as the best-so-far solution may not contain such information, as possible lower cardinality solutions might have significant differences or even be completely disjoint to many or all higher cardinality solutions (without redundant labels). This observation suggests an alternative solution construction process (C-II) which limits the number of constructed labels to the size of the best-so-far solution decreased by one. This approach intends to minimize the number of connected

components and at the same time maximize the number of additional arcs being represented within these fixed cardinality label sets. This strategy aims to increase the likelihood of ending up with connected feasible solutions after the addition of the last label. The overall construction process can also be performed in a combined way, i.e. one half of the ants constructing feasible, the other one infeasible ones (C-III).

To counteract stagnation we optionally perform pheromone smoothing, similar to the approach presented in [14]. Hence, we replace each $\tau(L', l)$ in Eq. (3) by

$$\tau(L', l) + \lambda' \cdot \left(\max_{l \in L \setminus L'} \tau(L', l) - \min_{l \in L \setminus L'} \tau(L', l) \right), \quad (5)$$

where λ' is controlling the amount of smoothing. It is initially set to zero, and in case of stagnation, indicated by no improvement within it_λ iterations, successively increased in steps of λ / it_λ until the maximum value λ is reached. If no improvement occurs for $2 \cdot it_\lambda$ iterations, we reinitialize pheromones entirely.

2.3 Pheromone Update

Pheromone update, which takes place after each iteration basically consists of two components: evaporation and deposition. Whereas pheromone evaporation provides a mean for escaping local optima and enables to direct the search towards other regions of the solution space, pheromone deposition is the main mean to guide the search process towards regions appearing attractive as a result of solutions created so far.

Pheromone evaporation is governed by parameter ρ, the evaporation rate. The update rule is given by

$$\tau_i \leftarrow (1 - \rho') \cdot \tau_i, \quad (6)$$

where index i refers to all elements of the pheromone vector or matrix respectively. For (P-I) and (P-III) we directly use parameter ρ for ρ', in the case of (P-II) we set $\rho' = \rho / |L|$ in order to obtain a comparable evaporation for each particular label.

Afterwards, pheromone deposition is performed based upon certain solutions. Various strategies do exist regarding to which solutions are selected for this purpose, which we leave unspecified for the moment. However, all update rules have in common, that an amount $\Delta L'$ is added to the pheromone values corresponding to the labels of solution L'. Regarding (P-I) the update is straightforwardly performed by

$$\tau_l \leftarrow \tau_l + \Delta L', \text{ for all } l \in L'. \quad (7)$$

As pheromone model (P-III) also accounts for the position at which a certain label has been added we have to consider L' as an ordered set and refer to the label constructed in step i by $L'[i]$. The pheromone update is then performed by

$$\tau_{il} \leftarrow \tau_{il} + \Delta L', \text{ for each } L'[i]. \quad (8)$$

For model (P-II) the update is performed in the following way:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta L', \text{ for all } i,\ j \in L'. \quad (9)$$

Following the approach of $\mathcal{MAX} - \mathcal{MIN}$ Ant System we introduce upper and lower bounds for the pheromone values $\tau^{\max}$ and $\tau^{\min}$. Pheromone values are initialized by the arithmetic mean of these two values.

The value $\Delta L'$ is calculated differently for the two proposed construction methods. If only feasible solutions are constructed, we only deposit pheromone for solutions having no more labels then the best-so-far solution.

In order to evaluate constructed candidate solutions we need to develop a function $f(L')$ which discriminates between solutions with equal $|L'|$. An evaluation function $f(L')$ can be built by considering function

$$h(L') = 1 - \frac{|E(L')|}{|E|}, \quad (10)$$

which accounts for additional edges being represented by labels L'. Feasible solutions can thus be evaluated by $f(L') = |L| + h(L')$.

If, on the other hand, infeasible solutions are constructed, we primarily have to account for the number of connected components induced by L'. We therefore use $f(L') = c(L') + h(L')$ in this case. In addition to the best-so-far feasible solution we also globally store the best-so-far created infeasible solution.

The pheromone deposition is performed by the best-so-far as well as the iteration-best ant in each iteration. If mixed construction (C-III) is performed, a total of four ants deposit pheromone. For both construction mechanisms we use $\Delta L' = 1$ when pheromone models (P-I) or (P-III) are used, and $\Delta L' = 1/|L'|$ in the case of model (P-III), which compensates the fact that for each label totally $|L'|$ pheromone values are increased. A further differentiation regarding the amount of pheromones to be deposited seems not to be reasonable within this context. In order to not implicitly limit pheromone values, we perform the evaporation step after the pheromone deposition.

2.4 Local improvement

Typically used neighborhoods for the MLST problem consist of the replacement of k labels within the current solution. Using $f(L') = |L'| + h(L')$ again allows for a better discrimination of solutions of equal cardinality. Regarding the solution construction process restricted to the cardinality of the incumbent solution minus one (see Section 2.2), we might also consider local search procedures working on infeasible solutions. Within such a process, solutions are evaluated by $f(L') = c(L') + h(L')$. Besides the specification of the size k of the neighborhood, we may consider various reinsertion strategies after having removed k labels. Completely traversing the whole neighborhood w.r.t. to some solution with already having k labels removed,

is impracticable, in particular for larger k. Hence we follow the strategy to consider all extension candidates for the first k' places, and add the remaining $k - k'$ labels following the greedy MVCA strategy, similar to the approach used in [7].

A further local improvement method consists of simply checking the labels for redundancy, by removing each label of the solution, and then test for connectivity. In particular for solutions which labels induce many additional edges, this method is more likely to be successful.

2.5 Implementation Aspects

In order to compute the number of components induced by a certain set of labels L' we use a *disjoint set* (also called *union find*) datastructure (see for instance [10]), as also suggested in [12]. If we are considering edge set E', all the operations on the union find data structure can be carried out in a total time of $O(|E'| \cdot \alpha(|E'|, |V|))$, where $\alpha(|E'|, |V|)$ denotes the inverse *Ackermann function*. For all reasonably occurring $|E'|$ and $|V|$ it holds, that $\alpha(|E'|, |V|)) \leq 4$. In comparison, a depth-first search (DFS) procedure would take time $O(|V| + |E'|)$. Within local search algorithms but also the MVCA-heuristic and therefore also the computation of the local information within the ACO algorithm, we often face the situation of tentatively adding some label for evaluation, and then removing it immediately. Hence we can further benefit from disjoint set datastructures supporting a rollback mechanism. The major benefit of this approach is that we can evaluate all further labels w.r.t. some considered partial solution in quasi-constant time. Let us further consider the situation where we want to remove some label l' from a partial solution L', which also occurs within the local search procedure. In this case we need to rebuild the disjoint set datastructure which takes $O(|E(L' \backslash l')| \cdot \alpha(|E(L' \backslash l')|, |V|))$ time. In this situation DFS allows for an incremental computation by first removing $|E(l')|$ edges from the graph, and then running DFS ($O(|V|+|E(L' \backslash l')|)$). However, compared to the time required to rebuild the disjoint set datastructure $O(|E(L' \backslash l')| \cdot \alpha(|E(L' \backslash l')|, |V|))$, this is no real drawback. Consequently we consider the disjoint set datastructure to be overall more appropriate for the given task.

3. Computational Results

Within comprehensive preliminary testing we determined a generally well-working configuration of the presented algorithmic components. Table 1 gives a summary of the determined parameter settings used for the subsequently presented computational results. Relatively high values of β are required, to give the labels mostly reducing the number of connected components a reasonable high chance of being selected. On average, in particular in the first construction steps, almost all labels will provide comparable reductions in the number of connected components, but higher reductions provide significant information that should be exploited.

Table 1: Parameter settings

Description	Parameter	Value
minimum pheromone value	$\tau^{\min}$	10^{-3}
maximum pheromone value	$\tau^{\max}$	10
pheromone-contribution	α	2
local-information-contribution	β	12
evaporation rate	ρ	0.1
pheromone-smoothing parameter	λ	0.2
iterations for pheromone smoothing	it_λ	20

For our computational experiments we used the instance set from [7]. All tests have been performed on a Intel Nehalem E5540 (2,53 GHz) CPU, under Linux with Kernel 2.6.31.

Table 2 shows our results obtained for the instances with $|V| = 200$ and $|V| = 500$. Results have been computed with the parameter settings listed in Table 1 and $it = 100$ iterations and $M = 20$ ants. Columns VNS contain the results of the variable neighborhood search presented in [7], being the best method therein. For a certain number of nodes, groups with various ratios of number of labels compared to the number of nodes as well as various graph densities $|E| = d \cdot \frac{|V| \cdot (|V|-1)}{2}$ exist. For each group ten different instances do exist. Reported objective values (column "obj.") and running times t_b at which the best solution was found, are average values over these ten instances. In columns ACO we report our results for 30 independent runs. Objective function values are listed in column "obj.", corresponding standard-deviations in column $\bar{\sigma}_{\text{obj}}$. By t_{avg} we denote the average total running times, and by t_{b} the average times at which the best solutions of the individual runs have been obtained. Corresponding standard-deviations are listed in colums $\bar{\sigma}_{\text{t}}$ and $\bar{\sigma}_{\text{tb}}$. With this particular parameter settings it is possible to obtain good solutions relatively fast, but however, in particular for the low density instances the average objective function values are generally higher than the ones obtained by VNS.

In Table 3 we report the results for various configurations of the ACO algorithm for a selected subset of low density graphs, i.e. the hardest instances within the sample. Again, we performed 30 independent runs for each instance and used the parameter settings from Table 1, but 200 iterations and $M = 50$ ants. If local search is applied (indicated in column "LS"), we set $it = 100$ and $M = 30$ to compensate for the longer computational time required for each local improvement. The best objective value obtained for each group of instances is highlighted in the table. Unfortunately it is not possible to draw a clear conclusion which pheromone model is overall superior. Model (P-II) yields the best results for instances with $|V| = 200,\ |L| \in \{100, 250\}$, but is generally worse for larger instances. For these instances (P-I) and (P-III) yield comparable results. The construction method (C-I) generally shows the worst performance on these instances,

Table 2: Results for instances with $|V| = 200$ and $|V| = 500$.

Parameters			VNS		ACO					
$\|V\|$	$\|L\|$	d	obj.	$t_{\mathrm{b}}[s]$	obj.	$\bar{\sigma}_{\mathrm{obj}}$	$t_{\mathrm{avg}}[s]$	$\bar{\sigma}_{\mathrm{t}}$	$t_{\mathrm{b}}[s]$	$\bar{\sigma}_{\mathrm{tb}}$
200	50	0.8	2.0	0.0	2.00	0.00	136.09	23.51	0.00	0.00
		0.5	2.2	0.03	2.20	0.00	15.21	6.59	0.07	0.26
		0.2	5.2	0.23	2.20	0.00	15.21	6.59	0.07	0.26
200	100	0.8	2.6	0.14	2.60	0.00	107.10	83.67	1.16	2.66
		0.5	3.4	0.16	3.40	0.00	23.48	7.93	2.41	5.07
		0.2	7.9	2.9	8.09	0.16	15.52	2.23	2.09	3.71
200	200	0.8	4.0	0.08	4.00	0.00	39.54	6.26	0.00	0.00
		0.5	5.4	0.88	5.40	0.00	33.56	5.10	7.58	8.25
		0.2	12.0	33.7	12.35	0.13	26.38	3.64	3.70	5.21
200	250	0.8	4.0	1.5	4.02	0.04	42.56	7.17	9.32	13.20
		0.5	6.3	2.3	6.37	0.05	35.87	5.34	4.53	6.48
		0.2	13.9	1.5	13.98	0.11	34.12	4.80	4.96	6.32
500	125	0.8	2	0.05	2.00	0.00	81.70	6.18	0.00	0.00
		0.5	2.6	0.56	2.61	0.01	109.32	18.48	6.10	20.25
		0.2	6.2	3.7	6.25	0.07	103.22	17.26	14.93	24.58
500	250	0.8	3	0.49	3.00	0.00	188.46	4.40	0.00	0.00
		0.5	4.1	26.9	4.26	0.08	196.18	57.65	7.52	32.97
		0.2	9.9	10.2	10.16	0.18	160.47	19.12	22.96	41.19
500	500	0.8	4.7	8.6	5.0	0.00	403.84	27.57	6.26	51.01
		0.5	6.5	110.2	7.30	0.20	365.20	47.80	20.02	70.72
		0.2	15.8	50.3	16.63	0.30	356.08	55.81	55.06	90.68
500	625	0.8	5.1	0.97	5.56	0.12	487.07	29.38	5.67	57.24
		0.5	7.9	33.9	8.39	0.08	629.85	62.70	41.04	101.34
		0.2	18.3	60.0	19.37	0.33	593.66	71.82	60.10	133.56

(C-II) and (C-III) show a similar average performance.

Various configurations regarding the subordinate local search method have been evaluated in preliminary experiments. However, due to longer running times, no configuration could outperform ACO without local improvement. To limit the time requirements for the local search, it turned out to be advantageous only to apply it for the best solutions regarding number of labels and number of components for feasible and infeasible solutions respectively. The neighborhood size was set to $k = 2$, as smaller neighborhoods did not yield sufficient improvements, and traversing larger neighborhoods turned out to be too time-consuming.

Although the running times of the configurations reported in Table 3 are higher than the ones with less iterations and ants reported in Table 2, they are still reasonable for many purposes. With these configurations improved average solution values compared to [7] could be obtained for some groups of instances.

4. Conclusions

Within this work we presented a description of the application of ant colony optimization to the minimum label spanning tree problem. Various pheromone models have been discussed, as well as different approaches to the solution construction steps of the artificial ants. Here, it turned out to be advantageous to also construct incomplete rather than feasible solutions. This particular idea has also been pursued to the introduced neighborhood structures. Computational results show the algorithm to be an attractive alternative to existing metaheuristic approaches. In particular when data instances do not contain single isolated optimums, but rather numerous optimal solutions, the method reliably finds the global optimum within relatively short computational time.

References

[1] T. Brüggemann, J. Monnot, and G. J. Woeginger. Local search for the minimum label spanning tree problem with bounded color classes. *Oper. Res. Lett.*, 31(3):195–201, 2003.

[2] M. Captivo, J. C. Clímaco, and M. M. Pascoal. A mixed integer linear formulation for the minimum label spanning tree problem. *Computers & Operations Research*, 36(11):3082 – 3085, 2009.

[3] R. Cerulli, A. Fink, M. Gentili, and S. Voß. Metaheuristics comparison for the minimum labelling spanning tree problem. In *Operations Research/Computer Science Interfaces Series*, volume 29, pages 93–106. Springer US, 2005.

[4] R.-S. Chang and S.-J. Leu. The minimum labeling spanning trees. *Information Processing Letters*, 63(5):277–282, 1997.

Table 3: Comparison of different pheromone models and construction mechanisms for instances with $|V| = 200$ and $|V| = 500$; $M = 50$, $it = 200$.

$\|V\|$	$\|L\|$	d	Pheromones	Construction	LS	obj.	$\bar{\sigma_{\text{obj}}}$	$t_{\text{avg}}[s]$	$\bar{\sigma_{\text{t}}}$	$t_{\text{b}}[s]$	$\bar{\sigma_{\text{tb}}}$
			P-I	C-I	–	8.17	0.07	80.49	12.43	3.68	11.98
			P-I	C-II	–	8.00	0.14	173.93	142.17	28.99	77.59
			P-I	C-III	–	8.03	0.15	75.30	14.18	7.56	17.24
			P-II	C-I	–	8.02	0.14	96.09	14.82	9.62	18.07
200	100	0.2	P-II	C-II	–	**7.91**	0.03	49.75	5.15	5.14	10.29
			P-II	C-III	–	7.93	0.06	75.03	9.95	8.80	17.01
			P-III	C-I	–	8.16	0.12	92.19	14.40	5.46	14.03
			P-III	C-II	–	7.95	0.10	108.51	76.15	17.86	40.03
			P-III	C-III	–	8.10	0.16	102.63	34.08	8.66	24.93
			P-III	C-I	✓	8.09	0.18	165.99	123.16	26.10	60.07
			P-I	C-I	–	12.27	0.07	116.88	14.70	6.62	13.35
			P-I	C-II	–	12.14	0.10	205.14	153.83	19.96	53.90
			P-I	C-III	–	12.19	0.13	103.74	27.96	8.63	18.18
			P-II	C-I	–	12.30	0.14	177.24	26.89	18.86	33.27
200	200	0.2	P-II	C-II	–	12.14	0.16	89.54	11.64	11.96	21.39
			P-II	C-III	–	12.24	0.18	136.87	18.49	11.70	24.25
			P-III	C-I	–	12.28	0.12	155.74	21.49	12.41	18.89
			P-III	C-II	–	**12.10**	0.10	113.32	55.58	11.01	22.86
			P-III	C-III	–	12.17	0.12	131.43	34.58	12.34	24.48
			P-III	C-I	✓	12.25	0.15	199.75	176.59	50.82	53.48
			P-I	C-I	–	13.90	0.02	132.10	20.27	6.08	8.82
			P-I	C-II	–	**13.89**	0.03	191.20	110.09	3.78	17.94
			P-I	C-III	–	13.90	0.02	114.24	17.90	3.74	7.65
			P-II	C-I	–	13.93	0.07	213.18	29.64	25.45	35.18
200	250	0.2	P-II	C-II	–	**13.89**	0.05	118.40	15.71	5.64	10.17
			P-II	C-III	–	**13.89**	0.03	166.75	22.85	10.68	19.93
			P-III	C-I	–	13.90	0.02	178.46	23.16	14.24	16.11
			P-III	C-II	–	13.90	0.00	133.80	39.43	3.96	3.07
			P-III	C-III	–	13.90	0.00	150.73	26.62	6.16	7.72
			P-III	C-I	✓	13.91	0.05	161.63	92.77	44.68	48.69
			P-I	C-I	–	6.23	0.05	491.23	81.82	14.91	27.67
			P-I	C-II	–	6.20	0.00	542.53	91.33	16.08	31.82
			P-I	C-III	–	6.20	0.00	466.21	74.85	18.61	37.56
			P-II	C-I	–	6.20	0.00	538.08	89.72	24.36	44.55
500	125	0.2	P-II	C-II	–	6.20	0.00	342.43	50.45	19.82	36.25
			P-II	C-III	–	6.20	0.00	487.29	86.41	26.15	50.40
			P-III	C-I	–	6.25	0.07	589.74	112.51	22.20	53.69
			P-III	C-II	–	6.21	0.03	537.69	256.29	29.22	82.07
			P-III	C-III	–	6.20	0.00	543.99	98.21	23.05	48.80
			P-III	C-I	✓	6.21	0.03	551.19	247.50	121.28	223.48
			P-I	C-I	–	9.93	0.12	494.51	77.73	53.44	91.76
			P-I	C-II	–	**9.87**	0.10	393.68	70.28	49.23	84.44
			P-I	C-III	–	9.89	0.10	445.86	67.57	59.85	102.39
			P-II	C-I	–	10.05	0.14	587.90	98.54	79.25	135.14
500	250	0.2	P-II	C-II	–	10.03	0.18	460.34	86.31	65.70	111.33
			P-II	C-III	–	10.05	0.19	491.29	82.24	73.31	126.40
			P-III	C-I	–	9.89	0.11	537.09	87.52	75.80	108.99
			P-III	C-II	–	9.91	0.10	384.20	68.47	47.06	70.65
			P-III	C-I	✓	10.03	0.18	612.77	289.21	190.16	267.34

[5] A. M. Chwatal, G. R. Raidl, and K. Oberlechner. Solving a k-node minimum label spanning arborescence problem to compress fingerprint templates. *Journal of Mathematical Modelling and Algorithms*, 8:293–334, 2009.

[6] S. Consoli, K. Darby-Dowman, N. Mladenovic, and J. Moreno-Pérez. Solving the minimum labelling spanning tree problem using hybrid local search. Technical report, n/a, 2007.

[7] S. Consoli, K. Darby-Dowman, N. Mladenovic, and J. Moreno-Pérez. Heuristics based on greedy randomized adaptive search and variable neighbourhood search for the minimum labelling spanning tree problem. *European Journal of Operational Research*, 196:440–449, 2009.

[8] S. Consoli, J. A. Moreno, N. Mladenovic, and K. Darby-Dowman. Constructive heuristics for the minimum labelling spanning tree problem: a preliminary comparison. Technical report, DEIOC Technical Report., 2006.

[9] S. Consoli, J. A. Moreno, N. Mladenovic, and K. Darby-Dowman. Mejora de la exploración y la explotación de las heurísticas constructivas para el mlstp. In *Spanish Meeting on Metaheuristics 2007*, 2007.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[11] M. Dorigo and T. Stützle. *Ant Colony Optimization*. The MIT Press, 2004.

[12] S. O. Krumke and H.-C. Wirth. On the minimum label spanning tree problem. *Information Processing Letters*, 66(2):81–85, 1998.

[13] J. Nummela and B. A. Julstrom. An effective genetic algorithm for the minimum-label spanning tree problem. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 553–558, New York, NY, USA, 2006. ACM.

[14] T. Stützle and H. H. Hoos. Max-min ant system. *Future Gener. Comput. Syst.*, 16(9):889–914, 2000.

[15] Y. Wan, G. Chert, and Y. Xu. A note on the minimum label spanning tree. *Information Processing Letters*, 84(2):99–101, 2002.

[16] Y. Xiong, B. Golden, and E. Wasil. Improved heuristics for the minimum label spanning tree problem. In *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, volume 10, December 2006.