Combining Lagrangian Decomposition with an Evolutionary Algorithm for the Knapsack Constrained Maximum Spanning Tree Problem

Sandro Pirkwieser[1], Günther R. Raidl[1], and Jakob Puchinger[2]

[1] Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
`{pirkwieser|raidl}@ads.tuwien.ac.at`
[2] National ICT Australia (NICTA) Victoria Laboratory
Dep. of Comp. Sci. & Softw. Eng., The University of Melbourne, Australia
`jakobp@csse.unimelb.edu.au`

Abstract. We present a Lagrangian decomposition approach for the Knapsack Constrained Maximum Spanning Tree problem yielding upper bounds as well as heuristic solutions. This method is further combined with an evolutionary algorithm to a sequential hybrid approach. Experimental investigations, including a comparison to a previously suggested simpler Lagrangian relaxation based method, document the advantages of the new approach. Most of the upper bounds derived by Lagrangian decomposition are optimal, and together with the evolutionary algorithm, large instances with up to 12000 nodes can be either solved to provable optimality or with a very small remaining gap in reasonable time.

1 Introduction

The *Knapsack Constrained Maximum Spanning Tree* (KCMST) problem has been introduced by Yamamato and Kubo [1]. It arises in practice in certain situations where the aim is to design a profitable communication network under a strict limit on total costs for cable laying or similar resource constraints.

We are given an undirected connected graph $G = (V, E)$ with node set V and edge set $E \subseteq V \times V$ representing all possible connections. Each edge $e \in E$ has associated a weight $w_e \in \mathbb{Z}_+$ (corresponding to costs) and a profit $p_e \in \mathbb{Z}_+$. In addition, a weight limit (capacity) $c > 0$ is specified. A feasible KCMST is a spanning tree $T \subseteq E$ on G, i.e. a cycle-free subgraph connecting all nodes, whose weight $\sum_{e\in T} w_e$ does not exceed c. The objective is to find a KCMST with maximum total profit $\sum_{e\in T} p_e$. More formally, we can introduce binary variables x_e, $\forall e \in E$, indicating which edges are part of the solution, i.e. $x_e = 1 \leftrightarrow e \in T$ and $x_e = 0$ otherwise, and write the KCMST problem as:

$$\max\ p(x) = \sum_{e\in E} p_e x_e \tag{1}$$

$$\begin{aligned} \text{s. t.} \quad & x \text{ represents a spanning tree} && (2)\\ & \sum_{e \in E} w_e x_e \leq c && (3)\\ & x_e \in \{0,1\} \qquad \forall e \in E && (4) \end{aligned}$$

Obviously, the problem represents a combination of the classical minimum spanning tree problem (with changed sign in the objective function) and the classical 0–1 knapsack problem due to constraint (3). Yamada et al. [2] gave a proof for the KCMST problem's $\mathcal{NP}$-hardness.

After summarizing previous work for this problem in the next section, we present a Lagrangian decomposition approach in Section 3. It is able to yield tight upper bounds as well as lower bounds corresponding to feasible heuristic solutions. Section 4 describes an evolutionary algorithm for the KCMST problem utilizing the edge-set representation. Section 5 explains how this evolutionary algorithm can be effectively combined with the Lagrangian decomposition approach in a sequential manner. Experimental results are presented in Section 6. They document the excellent performance of the whole hybrid system, which is able to solve almost all test instances with graphs of up to 12000 nodes to provable optimality or with a very small gap in reasonable time.

2 Previous Work

While numerous algorithms and studies exist for the standard minimum spanning tree problem, the 0–1 knapsack problem, and various related constrained network design problems, we are only aware of the following literature specifically addressing the KCMST problem.

Yamamato and Kubo [1] introduced this problem, but neither proved $\mathcal{NP}$-hardness nor presented any solution algorithms. This was first done by Yamada et al. [2]. They described a Lagrangian relaxation approach in which the knapsack constraint (3) is relaxed, yielding the simple maximum spanning tree problem which can be solved efficiently. The Lagrangian dual problem of finding a best suited Lagrangian multiplier for the relaxed weight constraint is solved by a simple bisection method. The Lagrangian relaxation approach also yields feasible heuristic solutions, which are further improved by a 2-opt local search. In order to also determine provable optimal solutions for instances of restricted size, the Lagrangian relaxation is embedded in a branch-and-bound framework. While the approach is able to optimally solve instances with up to 1000 nodes and 2800 edges when edge weights and profits are uncorrelated, performance degrades substantially in the correlated case.

The only other work for the KCMST problem we are aware of is the first author's master thesis [3]. It formed the basis for this article, and we refer to it for further details, in particular for more computational results.

The problem also exists in its minimization version [4], for which Jörnsten and Migdalas document the superiority of Lagrangian decomposition, and subsequently solving each subproblem to optimality, for generating valid bounds [5].

3 Lagrangian Decomposition for the KCMST Problem

Lagrangian relaxation is a commonly used technique from the area of mathematical programming to determine upper bounds for maximization problems. Though the solutions obtained are in general infeasible for the original problem, they can lend themselves to create feasible solutions and thus to derive lower bounds, too. For a general introduction to Lagrangian relaxation, see [6–8]. *Lagrangian Decomposition* (LD) is a special variant that can be meaningful when there is evidence of two or possibly more intertwined subproblems, and each of them can be efficiently solved on its own by specialized algorithms.

As the KCMST problem is a natural combination of the maximum spanning tree problem and the 0–1 knapsack problem, we apply LD by aiming at such a partitioning. For this purpose, we split variables x_e, $\forall e \in E$, by introducing new variables y_e and including linking constraints, leading to the following equivalent reformulation:

$$\begin{aligned}
\max\ & p(x) = \sum_{e \in E} p_e x_e && && (5)\\
\text{s. t.}\ & x \text{ represents a spanning tree} && && (6)\\
& \sum_{e \in E} w_e y_e \le c && && (7)\\
& x_e = y_e && \forall e \in E && (8)\\
& x_e, y_e \in \{0,1\} && \forall e \in E && (9)
\end{aligned}$$

The next step is to relax the linking constraints (8) in a Lagrangian fashion using Lagrangian multipliers $\lambda_e \in \mathbb{R}$, $\forall e \in E$. By doing so we obtain the Lagrangian decomposition of the original problem, denoted by KCMST-LD(λ):

$$\begin{aligned}
\max\ & p(x) = \sum_{e \in E} p_e x_e - \sum_{e \in E} \lambda_e (x_e - y_e) && && (10)\\
\text{s. t.}\ & x \text{ represents a spanning tree} && && (11)\\
& \sum_{e \in E} w_e y_e \le c && && (12)\\
& x_e, y_e \in \{0,1\} && \forall e \in E && (13)
\end{aligned}$$

Stating KCMST-LD(λ) in a more compact way and emphasizing the now independent subproblems yields

$$\begin{aligned}
\text{(MST)}\ \max\ & \{(p-\lambda)^T x \mid x \mathrel{\hat{=}} \text{a spanning tree},\ x \in \{0,1\}^E\}\ + && (14)\\
\text{(KP)}\ \max\ & \{\lambda^T y \mid w^T y \le c,\ y \in \{0,1\}^E\}. && (15)
\end{aligned}$$

For a particular λ, the maximum spanning tree (MST) subproblem (14) can be efficiently solved by standard algorithms. In our implementation we apply Kruskal's algorithm [9] based on a union-find data structure when the underlying graph is sparse and Prim's algorithm [10] utilizing a pairing heap with dynamic insertion [11] for dense graphs. The 0–1 knapsack subproblem (15) is known to be weakly $\mathcal{NP}$-hard, and practically highly efficient dynamic programming approaches exist [12], whereas we apply the COMBO algorithm [13].

It follows from Lagrangian relaxation theory that for any choice of Lagrangian multipliers λ, the optimal solution value to KCMST-LD(λ), denoted by v(KCMST-LD(λ)), is always at least as large as the optimal solution value of the original KCMST problem, i.e., KCMST-LD(λ) provides a valid upper bound. To obtain the tightest (smallest) upper bound, we have to solve the Lagrangian dual problem:

$$\min_{\lambda \in \mathbb{R}^E} v(\text{KCMST-LD}(\lambda)). \tag{16}$$

This dual problem is piecewise linear and convex, and standard algorithms like an iterative subgradient approach can be applied for (approximately) solving it. More specifically, we use the *volume algorithm* [14] which has been reported to outperform standard subgradient methods in many cases including set covering, set partitioning, max cut, and Steiner tree problems [15]. In fact, preliminary tests on the KCMST problem also indicated its superiority over a standard subgradient algorithm [3]. The volume algorithm's name is inspired by the fact that primal solutions are considered and that their values come from approximating the volumes below the active faces of the dual problem.

3.1 Strength of the Lagrangian Decomposition

According to integer linear programming theory, Lagrangian relaxation always yields a bound that is at least as good as the one obtained by the corresponding linear programming (LP) relaxation. The Lagrangian relaxation's bound can be substantially better when the relaxed problem does not fulfill the integrality property, i.e., the solution to the LP relaxation of the relaxed problem – KCMST-LD(λ) in our case – is in general not integer.

For seeing whether or not this condition is fulfilled here, we have to consider both independent subproblems. Compact models having the integrality property exist for MST, see e.g. [16]. Furthermore, the integrality property is obviously not fulfilled for the knapsack subproblem. Thus, we may expect to obtain bounds that are better than those from the linear programming relaxation of KCMST.

In comparison, in the Lagrangian relaxation approach from [2] the knapsack constraint is relaxed and only the MST problem remains. This approach therefore fulfills the integrality property and, thus, is in general weaker than our LD.

We further remark that the proposed LD can in principle be strengthened by adding the cardinality constraint $\sum_{e \in E} y_e = |V| - 1$ to the knapsack subproblem. The resulting cardinality constrained knapsack problem is still only weakly $\mathcal{NP}$-hard, and pseudo-polynomial algorithms based on dynamic programming are known for it [12]. Our investigations indicate, however, that the computational demand required for solving this refined formulation is in practice substantially higher and does not pay off the typically only small quality increase of the obtained bound [3].

3.2 Deriving Lower Bounds

In some iterations of the volume algorithm, the obtained spanning tree is feasible with respect to the knapsack constraint and can be directly used as a lower bound, hence resulting in a simple Lagrangian heuristic. In order to further improve such solutions this heuristic is strengthened by consecutively applying a local search based on the following edge exchange move.

1. Select an edge $(u, v) \in E \setminus T$ to be considered for inclusion (see below).
2. Determine the path $P \subseteq T$ connecting nodes u and v in the current tree. Including e in T would yield the cycle $P \cup \{(u, v)\}$.
3. Identify a least profitable edge $\tilde{e} \in P$ that may be replaced by (u, v) without violating the knapsack constraint:
$$\tilde{e} = \text{minarg } \left\{p_e \mid e \in E \wedge w(T) - w_e + w_{(u,v)} \leq c\right\}, \tag{17}$$
where $w(T) = \sum_{e \in T} w_e$. In case of ties, an edge with largest weight is chosen.
4. If replacing $\tilde{e}$ by (u, v) improves the solution, i.e. $p_{\tilde{e}} < p_{(u,v)} \vee (p_{\tilde{e}} = p_{(u,v)} \wedge w_{\tilde{e}} > w_{(u,v)})$, perform this exchange.

For selecting edge (u, v) in step 1 we consider two possibilities:

Random selection: Randomly select an edge from $E \setminus T$.

Greedy selection: At the beginning of the local search, all edges are sorted according to decreasing $p'_e = p_e - \lambda_e$, the reduced profits used to solve the MST subproblem. Then, in every iteration of local search, the next less profitable edge not active in the current solution is selected. This results in a greedy search where every edge is considered at most once.

Since Lagrangian multipliers are supposed to be of better quality in later phases of the optimization process, local search is only applied when the ratio of the incumbent lower and upper bounds is larger than a certain threshold τ. Local search stops after ρ consecutive non-improving iterations have been performed.

4 A Suitable Evolutionary Algorithm

Evolutionary algorithms (EAs) have often proven to be well suited for finding good approximate solutions to hard network design problems. In particular for constrained spanning tree problems, a large variety of EAs applying very different representations and variation operators have been described, see e.g. [17] for an overview.

Here, we apply an EA based on a direct edge-set representation for heuristically solving the KCMST problem, since this encoding and its corresponding variation operators are known to provide strong locality and heritability. Furthermore, variation operators can efficiently be applied in time that depends (almost) only linearly on the number of nodes. In fact, our EA closely follows the description of the EA for the degree constrained minimum spanning tree problem in [17]. Only the initialization and variation operators are adapted to conform with the knapsack constraint.

The general framework is steady-state, i.e. in each iteration one feasible offspring solution is created by means of recombination, mutation, and eventually local improvement, and it replaces the worst solution in the population. Duplicates are not allowed in the population; they are always immediately discarded. The EA's operators work as follows.

Initialization. To obtain a diversified initial population, a random spanning tree construction based on Kruskal's algorithm is used. Edges are selected with a bias towards those with high profits. The specifically applied technique is exactly as described in [17]. In case a generated solution is infeasible with respect to the knapsack constraint, it is stochastically repaired by iteratively selecting a not yet included edge at random, adding it to the tree, and removing an edge with highest weight from the induced cycle.

Recombination. An offspring is derived from two selected parental solutions in such a way that the new solution candidate always exclusively consists of inherited edges: In a first step all edges contained in both parents are immediately adopted. The remaining parental edges are merged into a single candidate list. From this list, we iteratively select edges by binary tournaments with replacement favoring high-profit edges. Selected edges are included in the solution if they do not introduce a cycle; otherwise, they are discarded. The process is repeated until a complete spanning tree is obtained. Finally, its validity with respect to the knapsack constraint is checked. An infeasible solution is repaired in the same way as during initialization, but only considering parental edges for inclusion.

Mutation. We perform mutation by inserting a randomly selected new edge and removing another edge from the introduced cycle. The choice of the edge to be included is biased towards high-profit edges by utilizing a normally-distributed rank-based selection as described in [17]. The edge to be removed from the induced cycle is chosen at random among those edges whose removal would retain a feasible solution.

Local Search. With a certain probability, a newly derived candidate solution is further improved by the local search procedure described in Section 3.2.

5 Hybrid Lagrangian Evolutionary Algorithm

Preliminary tests clearly indicated that the EA cannot compete with the performance of LD in terms of running time and solution quality. However, following similar ideas as described in [15] for the price-collecting Steiner tree problem, we can successfully apply the EA for finding better final solutions after performing LD. Hereby, the EA is adapted to exploit a variety of (intermediate) results from LD. In detail, the following steps are performed after LD has terminated and before the EA is executed:

1. If the profit of the best feasible solution obtained by LD corresponds to the determined upper bound, we already have an optimal solution. No further actions are required.
2. For the selection of edges during initialization, recombination, and mutation of the EA, original edge profits p_e are replaced by reduced profits $p'_e = p_e - \lambda_e$. In this way, Lagrangian dual variables are exploited, and the heuristic search emphasizes the inclusion of edges that turned out to be beneficial in LD.
3. The edge set to be considered by the EA is reduced from E to a subset E' containing only those edges that appeared in any of the feasible solutions encountered by LD. For this purpose, LD is extended to mark these edges.
4. The best feasible solution obtained by LD is included in the EA's initial population.
5. Finally, the upper bound obtained by LD is passed to the EA and exploited by it as an additional stopping criterion: When a solution with a corresponding total profit is found, it is optimal and the EA terminates.

6 Experimental Results

The described algorithms have been tested on a large variety of different problem instances, and comparisons have been performed in particular with the previous Lagrangian relaxation based method from [2]. This section summarizes most important results; more details can be found in [3]. All experiments were run on a 1.6GHz Pentium M PC with 1.25GB RAM.

As in [2], we consider instances based on random complete graphs $\mathrm{K}_{|V|\gamma}$ and planar graphs $\mathrm{P}_{|V|,|E|\gamma}$. Since we could not obtain the original instances, we created them in the same way by our own. In addition we constructed larger maximal planar graphs $\mathrm{P}_{|V|\gamma}$. Parameter γ represents the type of correlation between profits and weights:

uncorrelated ('u'): p_e and w_e, $e \in E$, are independently chosen from the integer interval $[1, 100]$;
weakly correlated ('w'): w_e is chosen as before, and $p_e := \lfloor 0.8 w_e + v_e \rfloor$, where v_e is randomly selected from $[1, 20]$;
strongly correlated ('s'): w_e is chosen as before, and $p_e := \lfloor 0.9 w_e + 10 \rfloor$.

For details on the methods used to construct the (maximal) planar graphs, we refer to [2, 3]. In case of complete graphs, the knapsack capacity is $c = 20 \cdot |V| - 20$, in case of (maximal) planar graphs $c = 35 \cdot |V|$. For each combination of graph type, graph size, and correlation, 10 instances have been considered.

We show and compare results for the Lagrangian relaxation (LR), Lagrangian relaxation with local search (LR+LS), and associated branch-and-bound (B&B) from [2], our Lagrangian decomposition with the simple primal heuristic (LD) and optionally local search (LD+LS), and the combination of LD and the EA (LD+LS+EA).

Robust settings for strategy parameters have been determined by preliminary tests. For the results presented here the following setup has been used.

The volume algorithm within the LD approach terminates when either the lower and upper bounds become identical and, thus, an optimal solution has been reached, or when the upper bound did not improve over the last 500 iterations in case of planar graphs and 1000 iterations in case of complete graphs. For completeness, we provide the following further details for the volume algorithm based on its description in [14]: The target value T always is updated by $T := 0.95LB$ and $T := 0.475(LB+UB)$ for planar and complete graphs, respectively, with the exception $T := 0.95T$ iff $UB < 1.05T$. Parameter f is initialized with 0.1 and multiplied by 0.67 after 20 consecutive *red* iterations when $f > 10^{-8}$ in case of planar graphs and $f > 10^{-6}$ for complete graphs and is multiplied by 1.1 after a *green* iteration when $f < 1$. Factor α is initialized with 0.1 and it is checked after every 100 and 200 iterations in case of planar and complete graphs, respectively, if the upper bound decreased less than 1%; if so and $\alpha > 10^{-5}$ then $\alpha := 0.85\alpha$. All these update rules are similar to those used in [15].

For the optional local search, greedy edge selection is used for complete graphs and random edge selection for all others. The application threshold is set to $\tau = 0.99$. As maximum number of iterations without improvement, $\rho = 200$ is used in case of uncorrelated and weakly correlated planar graphs, and $\rho = 100$ in all other cases.

For the EA, the population size is 100, binary tournament selection is used, and recombination and mutation are always applied. For the biasing towards edges with higher profits, parameters α and β (see [17]) are both set to 1.5. Local search is performed with random edge selection for each new candidate solution with a probability of 20% with $\rho = 50$ and a maximum of 5000 total iterations for graphs having less than 8000 nodes and 10000 total iterations for larger graphs.

Results on planar and complete graphs are shown in Table 1. For LR, LR+LS, and B&B, they are adopted from [2]. Average values based on 10 different instances are printed. Columns LB show obtained lower bounds, i.e. the objective values of the best feasible solutions. Upper bounds (UB) are expressed in terms of the relative gap to these lower bounds: $gap = (UB - LB)/LB$; corresponding standard deviations are listed in columns σ_{gap}. Columns *Opt* show numbers of instances (out of 10) for which the gap is zero and, thus, optimality has been proven. Average CPU-times for the runs are printed in columns t in seconds, and the average numbers of iterations of the volume algorithm in columns *iter*.

With respect to the CPU-times listed for branch-and-bound, we remark that they were measured on an IBM RS/6000 44P Model 270 workstation, and therefore, they cannot directly be compared with the times from our methods. The maximum time limit for B&B was 2000 seconds.

Most importantly, we can see that LD obtains substantially smaller gaps than both, LR and LR+LS. In fact, LD's average gaps are never larger than 0.063%, and for a large number of instances, optimality is already proven. On the remaining instances, enhancing LD by applying local search is beneficial; in most cases gaps are significantly reduced, and a few more instances could be solved to proven optimality. Overall, only 40 out of 330 instances remain, for

Table 1. Results of Lagrangian algorithms on planar and complete graphs.

Instance	**Yamada et al.**[2]				**LD**						**LD+LS**					
	LR	LR+LS	B&B													
	gap [$\cdot 10^{-5}$]	*gap* [$\cdot 10^{-5}$]	$t[s]$	*Opt*	$t[s]$	*iter*	*LB*	*gap* [$\cdot 10^{-5}$]	σ_{gap} [$\cdot 10^{-5}$]	*Opt*	$t[s]$	*iter*	*LB*	*gap* [$\cdot 10^{-5}$]	σ_{gap} [$\cdot 10^{-5}$]	*Opt*
$P_{50,127u}$	948.2	454.1	0.43	10	0.19	983	3558.5	62.56	89.70	3	0.30	976	3559.0	**47.58**	49.16	3
$P_{100,260u}$	586.6	268.9	1.78	10	0.17	801	7222.9	**6.76**	13.17	7	0.37	817	7222.9	**6.76**	13.17	7
$P_{200,560u}$	411.6	187.9	5.46	10	0.31	869	14896.7	3.98	5.60	6	0.55	822	14896.9	**2.68**	4.71	7
$P_{400,1120u}$	128.3	70.4	24.44	10	0.55	880	29735.0	2.71	3.83	6	1.15	905	29735.1	**2.36**	3.20	6
$P_{600,1680u}$	121.2	54.1	75.25	10	0.79	934	44836.2	1.11	1.17	5	1.52	854	44836.4	**0.67**	1.07	7
$P_{800,2240u}$	296.2	124.9	466.37	10	0.79	766	59814.5	**0**	0	10	1.59	716	59814.5	**0**	0	10
$P_{1000,2800u}$	166.0	73.3	592.77	10	0.99	764	74835.6	**0**	0	10	2.08	764	74835.6	**0**	0	10
$P_{50,127w}$	4372.0	1243.3	0.81	10	0.15	745	2063.2	52.80	79.75	6	0.23	751	2063.6	**33.57**	50.59	6
$P_{100,260w}$	2926.4	603.7	2.71	10	0.17	732	4167.9	9.67	16.94	7	0.36	724	4168.0	**7.24**	11.65	7
$P_{200,560w}$	1064.0	266.3	13.11	10	0.28	730	8431.9	1.19	3.76	9	0.36	634	8432.0	**0**	0	10
$P_{400,1120w}$	818.8	183.9	47.15	10	0.49	802	16794.3	3.58	6.42	7	0.77	721	16794.9	**0**	0	10
$P_{600,1680w}$	824.0	167.6	371.84	8	0.65	779	25158.0	**0.40**	1.26	9	1.29	788	25158.0	**0.40**	1.26	9
$P_{800,2240w}$	425.7	103.8	509.22	5	0.92	854	33540.2	0.89	1.99	8	1.76	762	33540.5	**0**	0	10
$P_{50,127s}$	10282.5	161.0	2.84	10	0.16	815	2051.3	43.92	62.81	5	0.12	573	2052.2	**0**	0	10
$P_{100,260s}$	19898.0	265.6	405.45	8	0.23	829	4115.1	9.72	12.54	6	0.18	641	4115.5	**0**	0	10
K_{40u}	250.9	106.1	0.87	10	0.23	880	3669.3	**5.50**	11.59	8	0.28	884	3669.3	**5.50**	11.59	8
K_{60u}	390.1	107.4	1.89	10	0.58	1164	5673.3	8.86	12.50	6	0.72	1189	5673.4	**7.10**	9.16	6
K_{80u}	272.7	130.3	6.54	10	0.60	858	7672.8	**0**	0	10	0.69	847	7672.8	**0**	0	10
K_{100u}	148.8	43.3	12.48	10	1.07	1062	9698.0	**1.03**	3.25	9	1.27	1055	9698.0	**1.03**	3.25	9
K_{120u}	122.3	42.7	23.69	10	1.37	1012	11701.2	**0**	0	10	1.65	1052	11701.2	**0**	0	10
K_{140u}	56.1	22.6	60.95	10	2.08	1184	13721.0	**0**	0	10	2.38	1162	13721.0	**0**	0	10
K_{160u}	89.7	38.8	476.26	10	2.88	1260	15727.9	**0**	0	10	3.19	1213	15727.9	**0**	0	10
K_{180u}	101.1	45.2	636.54	10	4.31	1488	17729.2	1.13	3.57	9	4.95	1470	17729.3	**0.56**	1.77	9
K_{200u}	40.5	17.2	375.26	10	5.55	1502	19739.4	**0**	0	10	6.11	1446	19739.4	**0**	0	10
K_{20w}	6186.9	991.7	0.25	10	0.11	720	618.9	**17.01**	53.79	9	0.12	698	618.9	**17.01**	53.79	9
K_{40w}	4262.5	520.3	1.17	10	0.24	737	1320.6	7.55	23.87	9	0.19	613	1320.7	**0**	0	10
K_{60w}	5700.5	529.2	6.09	10	0.51	891	2017.6	19.87	41.88	8	0.40	676	2018.0	**0**	0	10
K_{80w}	4970.4	343.6	38.15	10	0.81	863	2720.4	3.68	11.63	9	0.67	732	2720.5	**0**	0	10
K_{100w}	2413.3	172.9	377.61	8	1.10	879	3421.3	2.92	9.23	9	1.02	759	3421.4	**0**	0	10
K_{120w}	3797.7	206.6	451.06	8	2.78	1527	4123.3	26.69	24.15	3	1.65	871	4124.3	**2.43**	7.68	9
K_{20s}	22122.2	379.1	0.53	10	0.22	960	528.6	56.89	91.60	7	0.09	635	528.9	**0**	0	10
K_{30s}	17032.9	322.2	99.63	10	0.31	1016	809.2	37.12	59.76	7	0.16	717	809.5	**0**	0	10
K_{40s}	9492.7	137.7	226.30	6	0.34	902	1089.9	18.38	58.12	9	0.28	782	1090.1	**0**	0	10

which LD+LS was not able to find optimal solutions and prove their optimality. As already observed in [2], strongly correlated instances are typically harder to solve than uncorrelated ones.

A comparison of the heuristic solutions obtained from LD+LS with solutions from an exact approach[3] further indicated that almost all of them are actually optimal; LD+LS just cannot prove their optimality since the upper bounds were not tight enough. As a consequence, additionally applying the EA after LD+LS was not very meaningful for these instances. Tests not shown here confirmed that only in rare cases, gaps could further be reduced by the EA.

Our LD is extremely fast, needing for none of these instances more than seven seconds. The time overhead introduced by local search is also only very moderate, in particular since the improved heuristic solutions implied a faster convergence of the volume algorithm.

In order to investigate the usefulness of the proposed LD+LS+EA hybrid, we now turn to the larger maximal planar graphs, for which Table 2 presents

[3] We also implemented a not yet published exact branch-and-cut algorithm, which is able to solve these instances to proven optimality.

results. For the EA, we additionally list the average number of EA iterations $iter_{\mathrm{EA}}$, the relative amount of edges discarded after performing LD $red = (|E| - |E'|)/|E| \cdot 100\%$, and the number of optimal solutions Opt_{EA}, among Opt, found by the EA.

Again, the solutions obtained by LD are already quite good and gaps are in general small. The inclusion of local search clearly increases the number of optimal solutions found, leaving only 21 out of all 180 instances for which optimality is not yet proven. The hybrid approach (LD+LS+EA) works almost perfectly: Gaps are reduced to zero, and thus proven optimal solutions are achieved for all but three instances. The values in column Opt_{EA} document that the EA plays a significant role in finally closing gaps. The three remaining instances are solved with gaps less than 0.00003%.

In general, results of Tables 1 and 2 indicate that it is harder to close the optimality gap for smaller than for larger instances. One reason seems to be that with increasing graph size, more edges have the same profit and weight values. Tests on other types of instances, with differently determined profits and weights, are therefore interesting future work.

7 Conclusions

We presented a Lagrangian decomposition approach for the $\mathcal{NP}$-hard KCMST problem to derive upper bounds as well as heuristic solutions. Experimental results on large graphs revealed that the upper bounds are extremely tight, in fact most of the time even optimal. Heuristic solutions can be significantly improved by applying a local search, and many instances can be solved to provable optimality already in this way.

For the remaining, larger instances, a sequential combination of LD with an evolutionary algorithm has been described. The EA makes use of the edge-set encoding and corresponding problem-specific operators and exploits results from LD in several ways. In particular, the graph is shrunk by only considering edges also appearing in heuristic solutions of LD, Lagrangian dual variables are exploited by using final reduced costs for biasing the selection of edges in the EA's operators, and the best solution obtained from LD is provided to the EA as seed in the initial population.

Computational results document the effectiveness of the hybrid approach. The EA is able to close the gap and provide proven optimal solutions in almost all of the remaining difficult cases. Hereby, the increase in running time one has to pay is only moderate.

The logical next step we want to pursue is to enhance the branch-and-bound method from [2] by also utilizing the more effective LD or even the hybrid LD/EA instead of the simple Lagrangian relaxation.

In general, we believe that such combinations of Lagrangian relaxation and metaheuristics like evolutionary algorithms are highly promising for many combinatorial optimization tasks. Future work therefore includes the consideration

Table 2. Results of Lagrangian and hybrid algorithms on maximal planar graphs.

Instance	**LD**						**LD+LS**						**LD+LS+EA**								
	$t[s]$	$iter$	LB	gap $[\cdot 10^{-5}]$	σ_{gap} $[\cdot 10^{-5}]$	Opt	$t[s]$	$iter$	LB	gap $[\cdot 10^{-5}]$	σ_{gap} $[\cdot 10^{-5}]$	Opt	$t[s]$	$iter$	red	$iter_{\mathrm{EA}}$	LB	gap $[\cdot 10^{-5}]$	σ_{gap} $[\cdot 10^{-5}]$	Opt	Opt_{EA}
P_{2000u}	2.32	867	147799.4	0.14	0.29	8	3.26	813	147799.6	**0**	0	10	4.34	816	38%	2188	147799.6	**0**	0	10	1
P_{2000w}	2.42	883	85570.1	0.81	1.09	6	3.29	808	85570.7	0.12	0.37	9	6.29	856	44%	2001	85570.8	**0**	0	10	3
P_{2000s}	2.97	1045	82520.9	2.90	3.09	2	2.87	815	82523.3	**0**	0	10	3.33	816	20%	0	82523.3	**0**	0	10	0
P_{4000u}	4.64	854	294872.0	0.03	0.09	9	7.37	835	294872.0	**0.03**	0.09	9	12.42	853	39%	5000	294872.0	**0.03**	0.09	9	0
P_{4000w}	5.44	1040	170957.1	0.60	0.48	3	7.77	907	170957.7	0.24	0.50	8	12.93	985	43%	1283	170958.1	**0**	0	10	3
P_{4000s}	6.10	1071	165048.9	1.57	1.63	2	7.92	916	165051.4	0.06	0.18	9	7.53	887	23%	0	165051.5	**0**	0	10	0
P_{6000u}	8.16	953	441977.5	0.13	0.22	6	12.46	898	441978.1	**0**	0	10	23.45	959	39%	2674	441978.1	**0**	0	10	2
P_{6000w}	9.12	1033	256316.7	0.67	0.84	4	12.66	934	256318.3	0.04	0.12	9	20.93	980	45%	1130	256318.4	**0**	0	10	3
P_{6000s}	9.94	1094	247588.6	1.45	1.87	2	12.68	950	247592.2	**0**	0	10	15.62	937	25%	1325	247592.2	**0**	0	10	1
P_{8000u}	11.15	906	589446.9	0.04	0.08	8	19.58	975	589446.9	0.04	0.08	8	17.88	892	39%	0	589447.1	**0**	0	10	0
P_{8000w}	13.89	1102	341901.7	0.80	0.86	3	19.49	981	341904.0	0.12	0.20	7	26.97	919	46%	3503	341904.4	**0**	0	10	1
P_{8000s}	14.22	1087	330117.3	1.44	1.45	3	17.02	887	330122.0	**0.03**	0.09	9	39.37	922	23%	3968	330122.0	**0.03**	0.09	9	1
P_{10000u}	15.92	969	737450.2	0.07	0.12	7	24.63	956	737450.6	0.01	0.03	9	56.66	1029	39%	1877	737450.7	**0**	0	10	4
P_{10000w}	16.31	964	427406.4	0.19	0.39	7	25.51	1021	427406.9	0.06	0.09	7	61.62	1048	44%	1681	427407.2	**0**	0	10	5
P_{10000s}	23.42	1383	412640.1	0.84	0.87	1	26.61	1025	412643.6	**0**	0	10	26.82	1019	23%	0	412643.6	**0**	0	10	0
P_{12000u}	21.67	1056	885117.0	0.08	0.10	5	29.20	921	885117.8	**0**	0	10	55.54	1008	39%	1468	885117.8	**0**	0	10	3
P_{12000w}	23.27	1102	512985.4	0.38	0.48	3	32.69	1033	512986.9	0.08	0.13	7	77.05	1037	45%	2147	512987.3	**0**	0	10	4
P_{12000s}	25.83	1148	495164.0	1.14	1.38	2	34.38	1019	495169.5	0.04	0.08	8	141.99	1044	23%	8225	495169.6	**0.02**	0.06	9	1

of further problems, but also the closer investigation of other forms of collaboration between Lagrangian relaxation based methods and metaheuristics, including intertwined and parallel models.

References

1. Yamamato, Y., Kubo, M.: Invitation to the Traveling Salesman's Problem (in Japanese). Asakura, Tokyo (1997)
2. Yamada, T., Watanabe, K., Katakoa, S.: Algorithms to solve the knapsack constrained maximum spanning tree problem. Int. Journal of Computer Mathematics **82**(1) (2005) 23–34
3. Pirkwieser, S.: A Lagrangian Decomposition Approach Combined with Metaheuristics for the Knapsack Constrained Maximum Spanning Tree Problem. Master's thesis, Vienna University of Technology, Institute of Computer Graphics and Algorithms (October 2006)
4. Aggarwal, V., Aneja, Y., Nair, K.: Minimal spanning tree subject to a side constraint. Comput. & Operations Res. **9**(4) (1982) 287–296
5. Jörnsten, K., Migdalas, S.: Designing a minimal spanning tree network subject to a budget constraint. Optimization **19**(4) (1988) 475–484
6. Fisher, M.L.: The Lagrangian Relaxation Method for Solving Integer Programming Problems. Management Science **27**(1) (1981) 1–18
7. Fisher, M.L.: An application oriented guide to Lagrangean Relaxation. Interfaces **15** (1985) 10–21
8. Beasley, J.E.: Lagrangian relaxation. In Reeves, C.R., ed.: Modern Heuristic Techniques for Combinatorial Problems. John Wiley & Sons, Inc., New York (1993) 243–303
9. Kruskal, J.B.: On the shortest spanning subtree of a graph and the travelling salesman problem. In: Proc. of the AMS. Volume 7. (1956) 48–50
10. Prim, R.C.: Shortest connection networks and some generalizations. Bell Systems Technology Journal **36** (1957) 1389–1401
11. Fredman, M.L., Sedgewick, R., Sleator, D.D., Tarjan, R.E.: The pairing heap: a new form of self-adjusting heap. Algorithmica **1**(1) (1986) 111–129
12. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer Verlag (2004)
13. Martello, S., Pisinger, D., Toth, P.: Dynamic programming and strong bounds for the 0–1 knapsack problem. Management Science **45** (1999) 414–424
14. Barahona, F., Anbil, R.: The volume algorithm: producing primal solutions with a subgradient method. Mathematical Programming **87**(3) (2000) 385–399
15. Haouari, M., Siala, J.C.: A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem. Comput. & Operations Res. **33**(5) (2006) 1274–1288
16. Magnanti, T.L., Wolsey, L.A.: Optimal trees. In Ball, M.O., et al., eds.: Handbooks in Operations Research and Management Science. Volume 7. Elsevier Science (1995) 503–615
17. Julstrom, B.A., Raidl, G.R.: Edge sets: an effective evolutionary coding of spanning trees. IEEE Transactions on Evolutionary Computation **7**(3) (2003) 225–239