TECHNISCHE UNIVERSITÄT WIEN
Institut für Computergraphik und Algorithmen

A Lagrangian Decomposition/Evolutionary Algorithm Hybrid for the Knapsack Constrained Maximum Spanning Tree Problem

Sandro Pirkwieser and Günther R. Raidl and Jakob Puchinger

Forschungsbericht / Technical Report

TR–186–1–07–03

22. November 2007

Favoritenstraße 9-11 / E186, A-1040 Wien, Austria
Tel. +43 (1) 58801-18601, Fax +43 (1) 58801-18699
www.cg.tuwien.ac.at

A Lagrangian Decomposition/Evolutionary Algorithm Hybrid 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] NICTA Victoria Laboratory
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. Thorough experimental investigations, including a comparison to a previously suggested simpler Lagrangian relaxation based method, document the advantages of our approach. Most of the upper bounds derived by Lagrangian decomposition are optimal, and when additionally applying local search (LS) and combining it with the evolutionary algorithm, large and supposedly hard instances 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 arises in practice in situations where the aim is to design a profitable communication network under a strict limit on total costs, e.g. 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 $G_T = (V, T)$, $T \subseteq E$ 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,

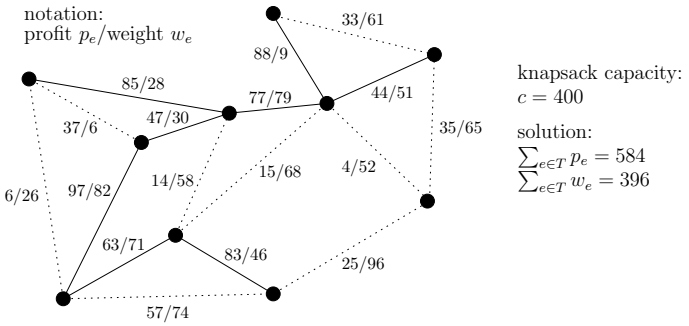

Fig. 1. Exemplary KCMST instance and its solution.

and write the KCMST problem as:

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

$$\text{s. t. } x \text{ represents a spanning tree on } G \tag{2}$$

$$\sum_{e \in E} w_e x_e \leq c \tag{3}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{4}$$

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

After summarizing previous work for this problem in the next section, we present a Lagrangian decomposition approach in Sect. 3. It is able to yield tight upper bounds as well as lower bounds corresponding to feasible heuristic solutions. The latter are gained via a Lagrangian heuristic including local search. 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. Computational results are presented in Sect. 6. The results document the excellent performance of the whole hybrid system, which is able to solve many test instances with planar graphs of up to 12000 nodes and complete graphs up to 300 nodes to provable optimality or with a very small gap in reasonable time.

This article extends our previous conference contribution [2] in various ways: more algorithmic details are presented, in particular concerning the volume algorithm for solving the Lagrangian dual; a new comparison of the Lagrangian decomposition with a previously proposed simpler Lagrangian relaxation is performed; and substantially more computational results for a larger variety of differently structured test instances are included.

2 Previous Work

In the literature, the KCMST problem is known under several different names and as minimization and maximization variants. As the minimization problem can trivially be transformed into a maximization variant, we ignore this difference in the following. Aggarwal et al. [3] were the first describing this problem and called it *MST problem subject to a side constraint*. They proved its $\mathcal{NP}$-hardness and proposed a branch-and-bound approach for solving it. Jörnsten and Migdalas [4] (*MST network subject to a budget constraint*) describe a *Lagrangian relaxation* (LR) in which the knapsack constraint (3) is relaxed, yielding a simple minimum spanning tree problem which can be solved efficiently. They further document the superiority of Lagrangian decomposition, and subsequently solving each subproblem to optimality, for generating valid bounds. An approximation algorithm also based on LR and a method to reduce the problem size is suggested in [5] (*constrained MST problem*). The later articles from Xue [6] (*weight-constrained MST*) and Jüttner [7] (*constrained minimum cost spanning tree problem*) deal with two similar primal-dual algorithms. Recently, Yamada et al. [1] (KCMST problem) also described a LR approach, which yields feasible heuristic solutions, too. These are further improved by a 2-opt local search. In order to determine provable optimal solutions for instances of restricted size, the LR 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. Our Lagrangian decomposition approach was introduced in the first author's master thesis [8]. Finally, the recent master thesis of Henn [9] gives an overview on previous work, introduces a way to reduce the problem size and presents another exact branch-and-bound scheme.

Generally, LR 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 LR, see [10–12].

Since LR plays a fundamental role in the mentioned previous work, we briefly present its straight-forward application to the KCMST problem. We denote it as KCMST-LR(λ):

$$\max \; p(x) = \sum_{e \in E} x_e(p_e - \lambda w_e) + \lambda c \tag{5}$$

$$\text{s. t. } \; x \text{ represents a spanning tree} \tag{6}$$

$$x_e \in \{0, 1\} \qquad \forall e \in E \tag{7}$$

In order to find a best suited Lagrangian multiplier $\lambda \geq 0$ for the relaxed weight constraint, one has to solve the Lagrangian dual problem:

$$\min_{\lambda \geq 0} v(\text{KCMST-LR}(\lambda)), \tag{8}$$

where the objective value of the optimal solution of KCMST-LR(λ) is denoted by v(KCMST-LR(λ)).

3 Lagrangian Decomposition for the KCMST Problem

Lagrangian decomposition (LD) is a special variant of LR 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 with the aim of such a partitioning. For this purpose, we duplicate variables x_e, $\forall e \in E$, by introducing new, corresponding variables y_e and including linking constraints, leading to the following equivalent reformulation:

$$\max \; p(x) = \sum_{e \in E} p_e x_e \tag{9}$$

$$\text{s. t. } \; x \text{ represents a spanning tree} \tag{10}$$

$$\sum_{e \in E} w_e y_e \leq c \tag{11}$$

$$x_e = y_e \qquad \forall e \in E \tag{12}$$

$$x_e, y_e \in \{0,1\} \qquad \forall e \in E \tag{13}$$

The next step is to relax the linking constraints (12) 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):

$$\max \; p(x) = \sum_{e \in E} p_e x_e - \sum_{e \in E} \lambda_e (x_e - y_e) \tag{14}$$

$$\text{s. t. } \; x \text{ represents a spanning tree} \tag{15}$$

$$\sum_{e \in E} w_e y_e \leq c \tag{16}$$

$$x_e, y_e \in \{0,1\} \qquad \forall e \in E \tag{17}$$

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

$$\text{(MST)} \; \max \; \{(p-\lambda)^T x \mid x \hat{=} \text{ a spanning tree}, x \in \{0,1\}^E\} \; + \tag{18}$$

$$\text{(KP)} \; \max \; \{\lambda^T y \mid w^T y \leq c, y \in \{0,1\}^E\}. \tag{19}$$

For a particular λ, the maximum spanning tree (MST) subproblem (18) can be efficiently solved by standard algorithms. In our implementation we apply Kruskal's algorithm [13] based on a union-find data structure when the underlying graph is sparse and Prim's algorithm [14] utilizing a pairing heap with dynamic insertion [15] for dense graphs. The 0–1 knapsack subproblem (19) is known to be weakly $\mathcal{NP}$-hard, and practically highly efficient dynamic programming approaches exist [16]; we apply the COMBO algorithm [17].

It follows from LR 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:

$$\min_{\lambda \in \mathbb{R}^E} v(\text{KCMST-LD}(\lambda)). \tag{20}$$

3.1 Solving the Lagrangian Dual Problem

The dual problem (20) 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* [18] which has been reported to outperform standard subgradient methods in many cases including set covering, set partitioning, max cut, and Steiner tree problems [19, 20]. Our preliminary tests on the KCMST problem also indicated its superiority over a standard subgradient algorithm [8]. 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. See Algorithm 1 for a pseudocode description.

The derived upper and lower bounds are stored in variables z_{UB} and z_{LB}, respectively. The primal vectors of the two subproblems, which represent an approximation to a primal solution, are denoted by $\boldsymbol{x}_{\text{P}}$ and $\boldsymbol{y}_{\text{P}}$, the Lagrangian multiplier vector is $\boldsymbol{\lambda}$.

At the beginning in line 1 an initial solution is created by solving the MST problem using edge values $v_e = p_e/w_e$, if this fails $v_e = 1/w_e$. In this way, either we derive a feasible solution or the problem instance is infeasible. In line 4 the Lagrangian multipliers are initialized to $\boldsymbol{\lambda}_e = 0.5p_e$. We remark that this as well as some other specific settings in the volume algorithm may influence the final solution quality significantly. Our choices are based on preliminary tests partly documented in [8] and the primary intention to find a relatively simple and generally robust configuration. The primal vectors are initialized in line 8. The target value T is always estimated by $T := 0.95z_{\text{LB}}$ with the exception $T := 0.95T$ if $z_{\text{UB}} < 1.05T$. Parameter f is initialized with 0.1 and multiplied by 0.67 after 20 consecutive *red* iterations (i.e. no better upper bound was found) when $f > 10^{-8}$ and is multiplied by 1.1 after a *green* iteration (i.e. a better upper bound was found and $\boldsymbol{v}^t \cdot (\boldsymbol{x}^t - \boldsymbol{y}^t) \geq 0$) when $f < 1$. These two parameters influence the step size, which determines the amount of change of the Lagrangian multipliers. Factor α controls the update of the primal vectors. It is initialized with 0.01 and periodically checked after every 100 iterations: if the upper bound decreased less than 1% and $\alpha > 10^{-5}$ then $\alpha := 0.85\alpha$. These initializations are done in line 9 and the update in line 31. The volume algorithm 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 300 iterations, i.e. $steps_{\max}$ is set to 300 in line 12. All these update rules are similar to those used in [20].

In each iteration the current subgradients $\boldsymbol{v}^t$, the step size s, and the new multipliers $\boldsymbol{\lambda}^t$ are determined. Using these multipliers both subproblems are

Algorithm 1: Volume Algorithm applied to KCMST

Result: best lower bound z_{LB}, best upper bound z_{UB} and best solution found sol_{best}

$(sol, p(sol)) \leftarrow$ `getInitialSolution()`;
$sol_{\text{best}} \leftarrow sol$;
$z_{\text{LB}} \leftarrow p(sol)$;
choose initial values for $\boldsymbol{\lambda}$;
$(z^0_{\text{MST}}, \boldsymbol{x}^0) \leftarrow$ `solve_MST(`$\boldsymbol{p} - \boldsymbol{\lambda}$`)`; `// see (18)`
$(z^0_{\text{KP}}, \boldsymbol{y}^0) \leftarrow$ `solve_KP(`$\boldsymbol{\lambda}$`)`; `// see (19)`
$z_{\text{UB}} = z^0_{\text{MST}} + z^0_{\text{KP}}$;
$(\boldsymbol{x}_{\text{P}}, \boldsymbol{y}_{\text{P}}) = (\boldsymbol{x}^0, \boldsymbol{y}^0)$; `// initialize primal values`
initialize T, f and α accordingly;
$t = 0$; `// iteration counter`
$steps = 0$;
while $z_{\text{LB}} \neq \lfloor z_{\text{UB}} \rfloor$ *and* $steps \neq steps_{\max}$ **do**
  $t = t + 1$;
  $\boldsymbol{v}^t = \boldsymbol{x}_{\text{P}} - \boldsymbol{y}_{\text{P}}$; `// determine actual subgradients`
  $s = f(z_{\text{UB}} - T)/||\boldsymbol{v}^t||^2$; `// determine step size`
  $\boldsymbol{\lambda}^t = \boldsymbol{\lambda} + s\boldsymbol{v}^t$; `// determine actual multipliers`
  $(z^t_{\text{MST}}, \boldsymbol{x}^t) \leftarrow$ `solve_MST(`$\boldsymbol{p} - \boldsymbol{\lambda}^t$`)`;
  $(z^t_{\text{KP}}, \boldsymbol{y}^t) \leftarrow$ `solve_KP(`$\boldsymbol{\lambda}^t$`)`;
  $z^t = z^t_{\text{MST}} + z^t_{\text{KP}}$; `// actual upper bound`
  `LagrangianHeuristic(`$\boldsymbol{x}^t$`)`; `// see Sect. 3.3`
  `// update` z_{LB} `and` sol_{best}
  $(\boldsymbol{x}_{\text{P}}, \boldsymbol{y}_{\text{P}}) = \alpha(\boldsymbol{x}^t, \boldsymbol{y}^t) + (1 - \alpha)(\boldsymbol{x}_{\text{P}}, \boldsymbol{y}_{\text{P}})$; `// update primal values`
  if $z^t < z_{\text{UB}}$ **then** `// better (lower) upper bound found`
    if $z^t < \lfloor z_{\text{UB}} \rfloor$ **then**
      $steps = 0$;
    else
      $steps = steps + 1$;
    $z_{\text{UB}} = z^t$; `// update best upper bound`
    $\boldsymbol{\lambda} = \boldsymbol{\lambda}^t$; `// update multipliers`
  else `// no improvement,` ***red*** `iteration`
    $steps = steps + 1$;
  update T, f and α accordingly;

solved and the upper bound z^t is calculated. Furthermore, a Lagrangian heuristic, described in Sect. 3.3, is applied to the solution of the MST subproblem, if necessary updating the lower bound and the best solution so far. Afterwards the primal values are updated using α; they are a convex combination of the preceding dual solutions $\boldsymbol{x}^0, \boldsymbol{y}^0$ to $\boldsymbol{x}^t$ and $\boldsymbol{y}^t$. Only in case a better (i.e. lower)

upper bound is found, the multipliers are set to the new values, and *steps* is reset to 0 iff $z^t < \lfloor z_{\mathrm{UB}} \rfloor$.

3.2 Strength of the Lagrangian Decomposition

According to integer linear programming theory, LR always yields a bound that is at least as good as the one obtained by the corresponding linear programming (LP) relaxation, providing the Lagrangian dual problem is solved to optimality. The LR'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.

To see whether or not this condition is fulfilled here, we have to consider both independent subproblems. For the MST problem, compact models having the integrality property exist, see e.g. [21]. For the knapsack problem, however, the integrality property is not fulfilled. Thus, we may expect to obtain bounds that are better than those from the linear programming relaxation of KCMST.

In comparison, in the LR approach from [1, 7] 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 or exact k-item knapsack problem is still only weakly $\mathcal{NP}$-hard, and pseudo-polynomial algorithms based on dynamic programming are known for it [16]. 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 improvement of the obtained bound [8].

3.3 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, we have already a trivial 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.
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:

$$\tilde{e} = \text{minarg } \left\{ p_e \mid e \in E \wedge w(T) - w_e + w_{(u,v)} \leq c \right\}, \tag{21}$$

   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. [22] 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 performed 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 [22]. 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 corresponds to that in [22]. 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 [22]. 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 Sect. 3.3.

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 [20] 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.

An outline of the collaboration is given in Fig. 2.

6 Computational Results

The described algorithms have been tested on a large variety of different problem instances, and comparisons regarding the strength of the Lagrangian dual have

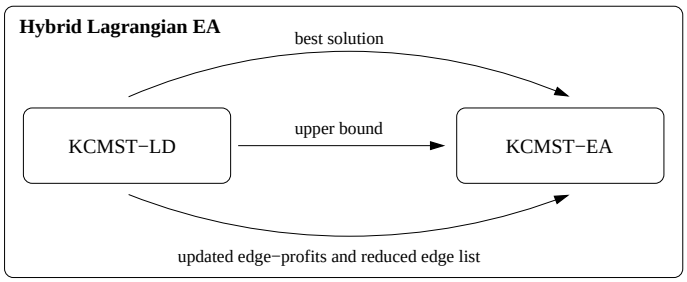

Fig. 2. Information exchange in the hybrid approach.

been performed in particular with the previous LR based primal-dual method of [7]. This section includes several representative results; further details can be found in [8]. All experiments were performed on a 2.2 GHz AMD Athlon 64 PC with 2 GB RAM.

We show and compare results for the Lagrangian relaxation (LR) based on [7], 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).

6.1 Test Instances

Unfortunately, no test instances from previously published algorithms for the KCMST problem are publicly available or could be obtained from the authors. As in [1], we consider instances based on complete graphs $\mathrm{K}_{|V|,\gamma}$ and planar graphs $\mathrm{P}_{|V|,|E|,\gamma}$. 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 $p_e := \lfloor 0.8w_e + v_e \rfloor$, where v_e is randomly selected from $[1, 20]$;

strongly correlated ("s"): w_e is chosen as before, and $p_e := \lfloor 0.9w_e + 10 \rfloor$.

For details on the methods used to construct the planar graphs, we refer to [1, 8]. 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 $\mathrm{P}_{|V|,\gamma}$, i.e. graphs that cannot be augmented by any further edge without violating planarity (for $|V| > 2 : \ |E| = 3|V| - 6$). In case of complete graphs, the knapsack capacity is $c = 20|V| - 20$, in case of (maximal) planar graphs $c = 35|V|$.

In particular for larger strongly correlated instances, we recognized that they are often easier to solve due to the relatively small number of possible profit and weight values and the resulting high probability for edges having assigned exactly the same profit/weight values. For example in case of our largest $\mathrm{P}_{8000,s}$ instances, there are 23994 edges but only 100 different profit/weight combinations. In the expected case this leads to ≈ 240 edges sharing each possible

profit/weight value pair. Therefore, we also created maximal planar graphs from a profit (weight) interval of $[1, 1000]$ and correspondingly scaled the correlations and the knapsack capacity. We denote these refined instances as $\mathrm{P}^*_{|V|,\gamma}$.

We further created particularly challenging test instances according to the description in [9]. They are based on random and complete graphs and the following special profit/weight correlations.

outliers ("o"): p_e and w_e lie with probability 0.9 in $[1001, 2000]$ and with probability 0.1 in $[1, 1000]$.

weakly correlated ("w_2"): p_e are uniformly distributed in $[1, 1000]$ and $w_e = \min\{1000, X + 0.5p_e\}$ with X chosen randomly in $[1, 1000]$.

strongly correlated ("s_2"): p_e are uniformly distributed in $[1, 1000]$ and $w_e = p_e + 20 + \beta$ with β uniformly in $[-20, 20]$.

To determine capacity c, the weight of the profit-maximal tree W_1 (in case of several such trees, the one having the least weight is chosen) and the weight of the weight-minimal tree W_2 are computed. Then c is derived in one of the following ways: $c = (W_1 + W_2)/4$ (low limit "l"), $c = (W_1 + W_2)/2$ (medium limit "m"), or $c = 3(W_1 + W_2)/4$ (high limit "h"). The variant used is given as additional subscript δ in the instance class name.

For each considered type, size, correlation, and capacity combination, 10 independent instances had been created.

6.2 Parameter Settings

In addition to the settings already described in Sect. 3.1 we are using the following setup for computing the results presented here.

For the optional local search, greedy edge selection is used for random and complete graphs with an application threshold set to $\tau = 0.99$ and random edge selection with $\tau = 0.995$ for the maximal planar graphs. In all cases $\rho = 100$ is used as maximum number of iterations without improvement. Heuristically derived solutions are not used for updating the target value T, thus the local search does not directly influence the volume algorithm.

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 [22]) are both set to 1.5. Local search is applied with a probability of 20% for each new candidate solution in the same manner as described before, except with $\rho = 50$. The maximum number of iterations is 10000 for (maximal) planar graphs and 30000 for random and complete graphs. In case of maximal planar graphs the edge set reduction was applied.

6.3 Comparing LR and LD

To see the performance differences between Lagrangian decomposition and the simpler Lagrangian relaxation, we compared our algorithm to a re-implementation

Table 1. Comparison between Lagrangian relaxation and decomposition.

Instance	Jüttner [7] LR				Our approach LD			
	$t[s]$	*iter*	%-gap_{L}	%-gap_{U}	$t[s]$	*iter*	%-gap_{L}	%-gap_{U}
$\mathrm{P}_{50,127,u}$	<0.01	6	0.4046	0.1349	0.05	805	0.0140	0.0478
$\mathrm{P}_{50,127,w}$	<0.01	6	1.0079	0.0485	0.09	704	0.0097	0.0291
$\mathrm{P}_{50,127,s}$	<0.01	4	4.3953	0	0.11	741	0.0487	0
$\mathrm{P}_{100,260,u}$	<0.01	7	0.2035	0.0249	0.10	726	0	0.0055
$\mathrm{P}_{100,260,w}$	<0.01	7	1.8282	0.0144	0.12	730	0.0072	0.0072
$\mathrm{P}_{100,260,s}$	<0.01	5	4.5438	0	0.16	746	0.0121	0
$\mathrm{K}_{20,u}$	<0.01	6	0.5369	0.2684	0.04	708	0.0061	0.0732
$\mathrm{K}_{20,w}$	<0.01	5	2.6822	0.1293	0.06	628	0.0485	0.0162
$\mathrm{K}_{20,s}$	<0.01	4	13.5186	0	0.08	723	0.0378	0
$\mathrm{K}_{40,u}$	<0.01	6	0.1935	0.0164	0.11	680	0.0055	0.0055
$\mathrm{K}_{40,w}$	<0.01	7	1.5371	0	0.23	721	0	0
$\mathrm{K}_{40,s}$	<0.01	4	5.6600	0	0.26	964	0.0459	0
$\mathrm{K}_{100,u}$	0.02	7	0.0454	0.0010	0.80	970	0	0.0010
$\mathrm{K}_{100,w}$	0.02	7	2.9257	0	1.32	978	0.0058	0
$\mathrm{K}_{100,s}$	0.01	4	5.7794	0	2.10	1529	0.0866	0
$\mathrm{R}_{100,1238,o,l}$	<0.01	8	0.2208	0.0429	5.79	2443	0.0039	0.0096
$\mathrm{R}_{100,1238,o,m}$	<0.01	8	0.0563	0.0068	0.87	1069	0.0012	0.0016
$\mathrm{R}_{100,1238,o,h}$	<0.01	6	5.8593	0.0007	0.34	784	0	0.0002
$\mathrm{R}_{100,1238,w_2,l}$	<0.01	9	0.5505	0.0413	2.72	1591	0.0036	0.0126
$\mathrm{R}_{100,1238,w_2,m}$	<0.01	9	0.1772	0.0143	1.11	1024	0.0050	0.0051
$\mathrm{R}_{100,1238,w_2,h}$	<0.01	8	0.0315	0.0065	0.48	865	0.0010	0.0005
$\mathrm{R}_{100,1238,s_2,l}$	<0.01	8	1.9856	0.0035	3.66	1063	0.0106	0.0020
$\mathrm{R}_{100,1238,s_2,m}$	<0.01	7	2.1569	0.0008	3.57	973	0.0045	0.0004
$\mathrm{R}_{100,1238,s_2,h}$	<0.01	8	0.3101	0.0005	3.22	979	0.0027	0.0003
avg. values	<0.01	6	2.3587	0.0314	1.14	964	0.0150	0.0090

of the method described in [7]. We made this choice since preliminary tests revealed that this method combines the good upper bounds of the bisection method in [1] and the good lower bounds of the primal-dual algorithm in [6] and, thus, outperforms both. Results on planar, complete, and random graphs are shown in Table 1; average values over 10 different instances are printed. Column $t[s]$ states the CPU-time in seconds, *iter* are the number of iterations. The table further lists relative errors of the achieved lower bounds %-$gap_{\mathrm{L}} = \frac{p^* - \underline{p}}{p^*} \cdot 100\%$ and those of the upper bound %-$gap_{\mathrm{U}} = \frac{\overline{p} - p^*}{p^*} \cdot 100\%$, with $\underline{p}$ and $\overline{p}$ being the derived lower and upper bounds, respectively, and the optimal solution value p^* was determined by an exact approach[3].

Most importantly, we can see that LD achieves in almost all cases substantially smaller gaps than LR and is never worse. In fact, LD's %-gap_{U} is never larger than 0.073% and %-gap_{L} is always below 0.087%, whereas the maxima of LR are ≈0.27% and even ≈13.5%, respectively. Thus, in the course of solving LD much more high-quality feasible solutions are derived. As already observed in [1],

[3] We also implemented a yet unpublished exact branch-and-cut algorithm, which is able to solve these instances to proven optimality.

strongly correlated instances are typically harder than uncorrelated ones, and Henn [9] also considered those with low capacity limit to be more challenging.

Sometimes LD is able to solve the instances to optimality but cannot prove their optimality since the upper bounds were not tight enough. In general, we can conclude that LD already delivers excellent bounds in short time.

6.4 LD combined with LS and EA

In order to investigate the performance of the proposed LD+LS and the LD+LS+EA hybrid, we turn to the larger maximal planar graphs, for which Table 2 presents results. Average values over 10 instances and 10 runs per instance (for the stochastic algorithms) are reported. If appropriate we state in the last row the average of these values over all instances.

We state again $t[s]$ and *iter*, but also the average lower bounds LB, 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 percentage of instances for which the gap is zero and, thus, optimality has been proven.

For LD+LS+EA, the table also lists the overall time $t[s]$, LB, corresponding gap information, the percentage of overall optimal solutions *%-Opt*, and additionally 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 percentage of optimal solutions $\%\text{-}Opt_{\mathrm{EA}}$, among *%-Opt*, found by the EA.

The solutions obtained by LD are already quite good and gaps are in general small. Applying the local search (LD+LS) always improves the average lower bound and in some cases helps to find more provably optimal solutions, which in turn reduces the number of iterations of the volume algorithm. The hybrid approach (LD+LS+EA) further boosts the average solution quality in almost all cases and substantially increases the numbers of solutions for which optimality could be proven. As expected, the finer-grained $\mathrm{P}^*_{|V|,\gamma}$ instances with larger profit and weight ranges are for all algorithms significantly harder to solve than the coarse-grained $\mathrm{P}_{|V|,\gamma}$ instances. The possible edge-set reduction decreases with increasing correlation and range. We remark that these large graphs are much harder to solve than the ones used in [1], thus the results are very satisfying; for LD+LS+EA, the gap is always less than 0.00023%.

Tests on random and complete graphs are shown in Table 3. The general results are quite similar than before, i.e. the local search as well as the EA are both consistently improving the quality. Preliminary tests suggested not to reduce the edge-sets on these type of instances; otherwise too many improving edges are missing. In comparison to the results presented by Henn [9], our approach was also highly successful on the challenging instances with outlier correlation (instances $\mathrm{R}_{|V|,|E|,o,\delta}$ and $\mathrm{K}_{|V|,o,\delta}$). In particular, LD+LS+EA was able to solve larger instances (300 instead of 200 nodes) to proven optimality or with a very small gap than could be tackled by Henn's branch-and-bound. We further solved nearly all strongly correlated graph instances to optimality (80 out of 90 with

Table 2. Results of Lagrangian decomposition 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	$t[s]$	red	$iter_{\text{EA}}$	LB	gap $[\cdot 10^{-5}]$	σ_{gap} $[\cdot 10^{-5}]$	$\%$-Opt	$\%$-Opt_{EA}
$P_{2000,u}$	1.48	791	147799.50	0.0683	0.2049	90	2.28	782	147799.55	0.0342	0.1489	95	2.90	41.21	150	147799.60	0	0	100	5
$P_{2000,w}$	1.52	853	85570.50	0.3519	0.7513	80	2.38	844	85570.63	0.1994	0.5261	86	4.26	42.61	457	85570.78	0.0235	0.1643	98	12
$P_{2000,s}$	2.12	1030	82521.70	1.9389	2.3118	40	2.66	868	82523.30	0	0	100	2.66	21.99	0	82523.30	0	0	100	0
$P_{4000,u}$	3.35	859	294872.00	0.0340	0.1019	90	5.59	841	294872.03	0.0238	0.0866	93	8.64	40.17	316	294872.10	0	0	100	7
$P_{4000,w}$	4.19	1053	170956.70	0.8195	0.9155	40	6.15	978	170957.79	0.1813	0.306	72	14.66	43.82	842	170958.06	0.0234	0.1147	96	24
$P_{4000,s}$	4.71	1066	165049.80	1.0300	0.8590	30	5.99	915	165051.44	0.0364	0.1439	94	9.95	19.92	410	165051.48	0.0121	0.0848	98	4
$P_{6000,u}$	5.66	912	441977.80	0.0680	0.1038	70	9.33	886	441977.96	0.0317	0.0786	86	15.41	40.25	339	441978.10	0	0	100	14
$P_{6000,w}$	6.55	1022	256317.40	0.3904	0.4621	50	9.25	964	256318.09	0.1210	0.2452	76	24.47	45.14	909	256318.36	0.0156	0.0764	96	20
$P_{6000,s}$	8.14	1157	247587.90	1.7368	1.3032	20	10.44	996	247592.04	0.0646	0.1481	84	33.73	19.94	1401	247592.09	0.0444	0.1264	89	5
$P_{8000,u}$	8.32	960	589446.50	0.1017	0.1357	60	13.81	918	589446.89	0.0356	0.077	81	28.44	39.98	595	589447.09	0.0017	0.0168	99	18
$P_{8000,w}$	9.78	1107	341902.50	0.5555	0.5139	30	14.18	1037	341903.85	0.1609	0.2124	58	48.40	44.82	1384	341904.37	0.0088	0.0499	97	39
$P_{8000,s}$	10.88	1125	330117.10	1.5147	1.3065	20	14.20	990	330121.86	0.0727	0.1294	76	57.00	17.99	1727	330121.96	0.0424	0.1051	86	10
$P^*_{2000,u}$	3.90	1189	1475226.30	0.2432	0.3076	20	5.74	1189	1475226.59	0.2236	0.2751	20	34.85	40.22	8000	1475226.93	0.2007	0.2466	20	0
$P^*_{2000,w}$	3.87	1151	854766.40	0.4685	0.5092	10	5.32	1151	854767.24	0.3698	0.3156	10	38.34	39.67	8906	854768.45	0.2281	0.1500	12	2
$P^*_{2000,s}$	4.80	1263	829462.20	0.7475	0.5877	10	6.17	1122	829468.13	0.0326	0.0535	73	16.61	4.98	2597	829468.15	0.0301	0.0522	75	2
$P^*_{4000,u}$	7.39	1230	2942682.80	0.1565	0.1262	30	11.17	1230	2942683.21	0.1425	0.1175	30	74.33	40.53	7000	2942684.72	0.0911	0.0812	30	0
$P^*_{4000,w}$	6.92	1179	1708837.70	0.6200	0.4678	20	9.89	1179	1708840.68	0.4457	0.3306	20	80.85	39.36	7852	1708845.20	0.1813	0.1454	23	3
$P^*_{4000,s}$	8.15	1288	1659023.60	0.5063	0.3548	0	11.92	1183	1659031.56	0.0265	0.0299	56	54.24	4.53	4320	1659031.59	0.0247	0.0296	59	3
$P^*_{6000,u}$	9.45	1178	4409459.00	0.0727	0.0827	30	15.22	1178	4409459.28	0.0664	0.0789	30	122.01	39.88	7000	4409460.15	0.0465	0.0445	30	0
$P^*_{6000,w}$	10.41	1256	2561360.20	0.4529	0.3598	0	15.40	1256	2561365.54	0.2444	0.1299	0	166.25	38.06	9683	2561369.06	0.1069	0.0799	9	9
$P^*_{6000,s}$	12.15	1335	2488583.60	0.4983	0.7021	10	18.10	1186	2488595.69	0.0125	0.0194	70	67.86	4.06	2967	2488595.70	0.0121	0.0193	71	1
$P^*_{8000,u}$	13.83	1290	5884437.60	0.0952	0.0556	0	22.51	1290	5884438.27	0.0838	0.0433	0	232.95	40.12	9667	5884439.85	0.0569	0.0354	5	5
$P^*_{8000,w}$	12.58	1189	3417468.90	0.1112	0.1093	40	18.76	1183	3417469.29	0.0998	0.1038	42	136.08	40.59	5414	3417471.21	0.0436	0.0565	49	7
$P^*_{8000,s}$	15.43	1298	3318159.40	0.4460	0.3934	20	25.90	1219	3318173.87	0.0099	0.0142	67	105.05	3.90	3246	3318173.89	0.0093	0.0139	69	2
avg. values	7.31	1115	–	0.5428	0.5427	33.7	10.93	1057	–	0.1132	0.1505	59.1	57.49	–	–	–	0.0501	0.0705	67.1	8.0

local search, and on average 86 out of 90 with the hybrid algorithm), which also documents that the derived upper bounds are in fact almost always optimal. In case of these graphs particularly the local search was highly effective. The remaining gap of LD+LS+EA is never worse than 0.0026%. In particular for $R_{300,22425,s_2,l}$ and $R_{300,22425,s_2,h}$ instances, our algorithm needed substantially less CPU time than [9][4].

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 and diverse 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 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 can be 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 always improves the quality and sometimes is able to close the gap and provide proven optimal solutions in many of the remaining difficult cases. Hereby, the increase in running time one has to pay is mostly only moderate.

The logical next step we want to pursue is to embed the LD or even the hybrid LD/EA into an exact branch-and-bound algorithm, similar to the one in [1] which makes use of the simple Lagrangian relaxation. Another possibility would be to employ the EA in an intertwined way with an exact method. This would permit us to compare the results with other exact methods in a more direct way.

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 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.

[4] They used a roughly comparable test environment, a 2x 86_64 AMD Opteron workstation.

Table 3. Results of Lagrangian decomposition and hybrid algorithms on random and complete graphs with range 1000.

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	$t[s]$	$(red)^*$	$iter_{\mathrm{EA}}$	LB	gap $[\cdot 10^{-5}]$	σ_{gap} $[\cdot 10^{-5}]$	$\%$-Opt	$\%$-Opt_{EA}
$R_{300,11213,o,l}$	9.53	1737	542839.40	1.7477	1.8326	10	11.72	1737	542840.60	1.5271	1.5937	10	29.99	92.93	27000	542843.63	0.9706	0.6928	10	0
$R_{300,11213,o,m}$	7.10	1536	580716.50	0.2583	0.2464	30	8.89	1506	580716.60	0.2411	0.2576	40	21.43	91.63	18000	580716.64	0.2342	0.2477	40	0
$R_{300,11213,o,h}$	3.57	1260	591409.00	0.1690	0.2507	50	5.11	1259	591409.30	0.1183	0.1320	50	13.73	91.02	12285	591409.54	0.0778	0.1132	64	14
$R_{300,11213,s_2,l}$	24.58	1563	77466.60	8.5209	5.6046	20	24.45	1409	77473.00	0.2581	0.5161	80	24.69	80.64	336	77473.20	0	0	100	20
$R_{300,11213,s_2,m}$	15.37	1351	155244.80	5.4064	5.1165	0	14.77	1051	155253.20	0	0	100	14.73	81.54	0	155253.20	0	0	100	0
$R_{300,11213,s_2,h}$	16.52	1332	232877.70	6.5305	5.2668	10	16.74	1238	232892.50	0.1718	0.2847	70	18.34	85.28	2222	232892.89	0.0043	0.0428	99	29
$R_{300,22425,o,l}$	26.39	3324	568771.90	6.8383	6.1475	10	32.10	3324	568788.80	3.8714	4.3327	10	52.08	95.24	26700	568796.00	2.6042	3.3654	11	1
$R_{300,22425,o,m}$	14.70	1943	588410.30	0.2210	0.2020	30	18.83	1943	588410.50	0.1870	0.1605	30	33.05	95.46	18078	588410.80	0.1360	0.1272	40	10
$R_{300,22425,o,h}$	7.28	1358	594373.50	0.0168	0.0505	90	10.10	1358	594373.50	0.0168	0.0505	90	12.40	94.54	3000	594373.50	0.0168	0.0505	90	0
$R_{300,22425,s_2,l}$	44.08	2059	77445.70	12.2628	9.0170	0	42.58	1793	77455.20	0	0	100	42.58	86.26	0	77455.20	0	0	100	0
$R_{300,22425,s_2,m}$	29.69	1687	154940.30	7.8185	8.9007	10	28.81	1392	154952.40	0	0	100	28.81	93.71	0	154952.40	0	0	100	0
$R_{300,22425,s_2,h}$	34.63	1964	232424.80	16.2741	12.5659	10	36.55	1885	232461.90	0.3013	0.3874	50	44.59	89.39	10682	232462.37	0.0990	0.1811	77	27
$K_{300,o,l}$	247.29	19163	582646.00	4.0334	7.1749	10	316.33	19163	582660.30	1.5789	1.4435	10	333.98	97.50	27000	582663.46	1.0366	0.8511	10	0
$K_{300,o,m}$	40.44	2909	592797.70	0.1856	0.1401	30	45.96	2864	592797.90	0.1518	0.1401	40	55.19	97.70	10212	592798.50	0.0506	0.0773	70	30
$K_{300,o,h}$	30.13	2373	596076.40	0.0503	0.1074	80	35.49	2371	596076.50	0.0336	0.0671	80	36.13	96.94	1239	596076.70	0	0	100	20
$K_{300,s_2,l}$	63.20	2495	77225.70	28.6269	20.8442	0	60.80	2195	77247.80	0	0	100	60.80	93.07	0	77247.80	0	0	100	0
$K_{300,s_2,m}$	62.25	2704	154445.00	12.4958	8.3394	0	59.11	2404	154464.30	0	0	100	59.11	94.48	0	154464.30	0	0	100	0
$K_{300,s_2,h}$	76.60	3396	231665.00	15.9285	18.7408	10	78.10	3142	231701.90	0	0	100	78.10	92.77	0	231701.90	0	0	100	0
avg. values	41.85	3008	–	7.0769	6.1415	22.0	47.02	2890	–	0.4698	0.5203	64.0	53.3183	–	–	–	0.2905	0.3193	72.8	8.3

*No edge-set reduction applied, only states possible amount.

Acknowledgements

The Institute of Computer Graphics and Algorithms is supported by the European RTN ADONET under grant 504438, by the Austrian Exchange Service, Acciones Integradas Austria/Spain, under grant 13/2006 and by the Austrian Science Fund (FWF) under contract number P20342-N13.

NICTA is funded by the Australian Government's Backing Australia's Ability initiative, in part through the Australian Research Council.

References

1. 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
2. Pirkwieser, S., Raidl, G.R., Puchinger, J.: Combining Lagrangian decomposition with an evolutionary algorithm for the knapsack constrained maximum spanning tree problem. In Cotta, C., van Hemert, J., eds.: Evolutionary Computation in Combinatorial Optimisation - EvoCOP 2007. Volume 4446 of LNCS., Springer (2007) 176–187
3. Aggarwal, V., Aneja, Y., Nair, K.: Minimal spanning tree subject to a side constraint. Comput. & Operations Res. **9**(4) (1982) 287–296
4. Jörnsten, K., Migdalas, S.: Designing a minimal spanning tree network subject to a budget constraint. Optimization **19**(4) (1988) 475–484
5. Ravi, R., Goemans, M.X.: The constrained minimum spanning tree problem (extended abstract). In: SWAT '96: Proceedings of the 5th Scandinavian Workshop on Algorithm Theory. Volume 1097 of LNCS., London, UK, Springer-Verlag (1996) 66–75
6. Xue, G.: Primal-dual algorithms for computing weight-constrained shortest paths and weight-constrained minimum spanning trees. In: IEEE International Performance, Computing & Communications Conference. (2000) 271–277
7. Jüttner, A.: On resource constrained optimization problems. In: 4th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications. (2005)
8. 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)
9. Henn, S.T.: Weight-constrained minimal spanning tree problem. Master's thesis, University of Kaiserslautern, Department of Mathematics (May 2007)
10. Fisher, M.L.: The Lagrangian Relaxation Method for Solving Integer Programming Problems. Management Science **27**(1) (1981) 1–18
11. Fisher, M.L.: An application oriented guide to Lagrangean Relaxation. Interfaces **15** (1985) 10–21
12. 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
13. 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
14. Prim, R.C.: Shortest connection networks and some generalizations. Bell Systems Technology Journal **36** (1957) 1389–1401

15. 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
16. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer Verlag (2004)
17. Martello, S., Pisinger, D., Toth, P.: Dynamic programming and strong bounds for the 0–1 knapsack problem. Management Science **45** (1999) 414–424
18. Barahona, F., Anbil, R.: The volume algorithm: producing primal solutions with a subgradient method. Mathematical Programming **87**(3) (2000) 385–399
19. Bahiense, L., Barahona, F., Porto, O.: Solving steiner tree problems in graphs with lagrangian relaxation. Journal of Combinatorial Optimization **7**(3) (2003) 259–282
20. 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
21. 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
22. 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