

Solving the Extended Tree Knapsack Problem (ETKP) with fixed cost flow expansion functions

David J. van der Merwe*, Johannes M. Hattingh

Abstract— Parts of the LATN (Local Access Telecommunication Network) planning problem can be modelled with the Extended Tree Knapsack formulation. The LATN can constitute up to 60% of total network costs. In this paper a partitioning algorithm using standard off the shelf software coupled with enhanced modeling is presented. Enhancements to the algorithms is presented along with empirical results.

Index Terms—Network Design Considerations, Network Planning

I. INTRODUCTION

THE Local Access Telecommunication Network (LATN) connects individual subscribers to telecommunications networks through switching centres [1]. Generally telecommunication networks are hierarchically divided into two [2] or three parts [1], with the local access network contributing a large portion of the total network development cost, up to 60% according to [1] in the case fixed line situations.

Part of the LATN expansion problem can be modelled as an Extended Tree Knapsack Problem (ETKP) [15]. Expansion may be required when subscriber numbers have increased or if the demand for service of existing subscribers increase beyond the current capacity of the existing network. This problem has been treated extensively in [1], [2], [6], where models and solution strategies for the problem have been given. In many cases LATN design and expansion problems are divided into more than one step. In some of these cases the Tree Knapsack Problem (TKP) and Extended Tree Knapsack Problem (ETKP) can be used as model. To solve the overall design problem, it may be necessary to solve many TKP and ETKP models. It is thus important to be able to solve these problems efficiently, as these models may be solved a large number of times.

The focus of this paper is to present a solution strategy for the ETKP model with a fixed charge cost function. Previous work published in [19] published algorithms for the TKP

This work forms part of the research done at North-west University, Potchefstroom Campus within the TELKOM CoE research programme, funded by TELKOM, SAAB GRINTEK and THRIP.

Manuscript received May 3, 2008.

D. J. van der Merwe is with the North-west University, Potchefstroom Campus (phone: +27 18 299 2536; e-mail: David.vanderMerwe@nwu.ac.za).

J.M. Hattingh is with the North-west University, Potchefstroom Campus (e-mail: Giel.Hattingh@nwu.ac.za).

problem. The ETKP model is more complex resulting in more computational burden. Some results and algorithmic approaches have been discussed in [19]. In this paper we focus more on the ETKP and discuss new research that uses valid inequalities and improved lower bounds. The enhancements to the algorithm and is discussed in detail in the Appendix. Performance results are given in section VI.

II. BACKGROUND

Before discussing the ETKP model, a brief note on the general knapsack problem or 0-1 Knapsack Problem (KP) is presented. The KP is a widely studied problem in Operations Research and combinatorial optimization. The reason is that it has many variants and is found as subproblem for many optimization problems. Generally in knapsack problems a set of items are available with associated values. A subset of items must be chosen to maximize the total value, however, generally some sort of bound is imposed making it impossible to select all items from the original set. These types of problems get their name from the situation where a hitchhiker or soldier has to fill up his knapsack with items of value from a given set while keeping within some constraint. This constraint can be weight or volume or some other measure [11].

In the design of LATN networks a special type of knapsack problem is encountered where a precedence order is imposed on the items to be chosen. This means that in order to take a specific item, it is required that another item also be taken. For example, before a flash light can be included by a hiker, batteries must be included. After batteries have been included, it may be possible to pack a radio as well. The reverse is also true, if the batteries are taken out, so too must the radio and flashlight. The TKP and ETKP are discussed in [3], [4], [14] and [15].

In the following section the ETKP model will be presented.

III. ETKP MODEL

The ETKP model investigated can be seen as an extension to the TKP problem, as reported in [18] and [19]. In the following discussion the items that can be included in the ETKP can be seen as customers that can possibly be serviced and can alternatively be referred to as nodes. The ETKP can be seen as choosing a subtree from a tree that maximizes the profit for including nodes in the subtree while not violating some capacity constraint.

Given is an undirected tree $T = (V, E)$ rooted at node 0 where $V = \{0, 1, 2, \dots, n\}$ is the set of customers or nodes and E is the set of potential links between the nodes. For

each node $i \in V$ there exists a positive integer c_i representing the profit obtained for serving node j and a positive integer d_i representing the cost or capacity used up by serving node j . Two vital assumptions for the problem are

- 1) Indivisible demand assumption: Each node is serviced completely, meaning all its demand is serviced or not at all
- 2) Contiguity constraint: All nodes on the path between a specific node and the root node, must also be included if a node is included in the subtree.

The root node has limited capacity to serve the demand of the customers, this will be denoted by a non-negative integer H . Each node also generate a flow y_j , with an associated cost of $f(y_j)$. This cost function can take on several forms. In order to formulate the problem as a MILP certain modeling changes was implemented to the ETKP as presented in [15].

The vector \mathbf{x} has 0-1 elements x_j denoting the exclusion or inclusion of a node j in the subtree.

The y_j variables are divided into two parts, such that $y_j = y_{j1} + y_{j2}$, where y_{j1} is the part of the flow less than the current capacity ($y_{j1} \leq b_j$) and y_{j2} the part of the flow greater than the current capacity of the link.

Secondly, introduce new 0-1 variables that correspond to the expansion decisions, $\delta_j, j=1,2,3,\dots,n-1$ and logical constraints of the form $y_{j2} \leq H\delta_j$ to force y_{j2} to zero if $\delta_j = 0$.

Define the vectors as $\mathbf{y}_1^t = (y_{11}, y_{21}, \dots, y_{(n-1)1})$ and $\mathbf{y}_2^t = (y_{12}, y_{22}, \dots, y_{(n-1)2})$

The ETKP can now be formulated as the following MILP:

$$\max \sum_{j=0}^{n-1} c_j x_j - \sum_{j=1}^{n-1} (F_j \delta_j + \delta_j y_{j2}) \quad (1)$$

$$\text{s.t. } x_j - x_{p_j} \leq 0 \quad j=1,2,\dots,n-1 \quad (2)$$

$$x_0 = 1 \quad (3)$$

$$D\mathbf{x} - B(\mathbf{y}_1 + \mathbf{y}_2) = \mathbf{0}, \quad (4)$$

$$\mathbf{d}^T \mathbf{x} \leq H, \quad (5)$$

$$y_{j2} \leq H\delta_j, \quad j=1,2,\dots,n-1 \quad (6)$$

$$0 \leq y_{j1} \leq b_j \quad j=1,2,\dots,n-1 \quad (7)$$

$$y_{j1}, y_{j2} \geq 0 \quad j=1,2,\dots,n-1, \quad (8)$$

$$x_j \in \{0,1\} \quad j=0,1,\dots,n-1, \quad (9)$$

$$\delta_j \in \{0,1\} \quad j=1,2,3,\dots,n-1. \quad (10)$$

where p_j is the parent of node j . D is a diagonal matrix with diagonal elements d_1, d_2, \dots, d_n . The set of constraints $D\mathbf{x} - B\mathbf{y} = \mathbf{0}$ ensures the conservation of flow, where B represents the node-arc incidence matrix. The profit obtained from choosing a subtree can be seen as the profit for including nodes minus the cost of flow generated by the nodes chosen.

In this paper a specific form of the cable expansion cost function $f_j(y_j)$ is used.

$$f_j(y_j) = \begin{cases} 0, & \text{if } y_j \leq b_j \\ F_j + a_j(y_j - b_j), & \text{otherwise.} \end{cases} \quad (11)$$

where b_j is the current capacity of the link between nodes j and p_j , a_j is the variable cost for flow generated exceeding b_j and F_j is the fixed cost for sending flow in excess of b_j from node j to node p_j .

This cost function represents the case where no cost is incurred for sending flow less than or equal to b_j , from node j to p_j . Sending more than b_j units of flow incurs a fixed cost of F_j and a variable cost of a_j per extra unit of flow.

The next section will discuss general solution strategies and present the solution strategy used for the research forming the basis for this paper.

IV. GENERAL SOLUTION STRATEGIES

Various solution strategies exist for solving optimization problems, like standard off the shelf software, heuristics, dynamic programming, tabu search etc. For the TKP and ETKP models various dynamic programming and branch and bound methods are available, see [3], [4], [14] and [15]. More approaches for the general planning problem is presented in [1].

Specifically for the ETKP a depth-first dynamic programming algorithm is presented in [15]. Note that the ETKP is represented in a depth first manner for the implementation of this algorithm

The depth-first dynamic programming algorithm for the ETKP uses recursive rules to build solutions for sub trees and to generate an optimal solution value for the problem in a recursive fashion. A negative aspect is that once an optimal objective function has been obtained, the optimal solution sub-tree and flow dimensioning is not known. An additional procedure is presented in [15] that can be used to obtain the optimal solution, after the optimal objective function value has been determined.

V. ALGORITHM DEVELOPED

The solution approach used for the ETKP is a method that uses standard off the shelf software combined with enhanced modeling and partitioning. The problems are modeled as Mixed Integer Linear Programs (MILP). This approach has the following advantages:

- Custom developed solution algorithms (like dynamic programming) is seldom readily available for use by other practitioners. Using standard software will cut down on development times.
- Guarantees on the quality of solutions. Many dynamic programming solutions can give no guarantee regarding the quality of solutions obtained in cases where time constraints prevent it from running to completion.
- The solution process proposed here will create interim feasible solutions with known bounds; this is preferred to methods that only produce solutions upon completion.
- Advances in the standard software as vendors improve their product will improve the efficiency of the solution process automatically.

The algorithm developed produces exact optimal solutions, however, intermediate solutions are also generated during the solution process. These solutions form bounds for the overall problem and these bounds improves the performance of the solution process.

A two-phase partitioning of the search space is done. The first partition aims to exploit the idea of cardinality. The idea of cardinality is estimate the optimal number of variables in the optimal solution and use this estimate to reduce the search space. The idea of cardinality was advocated in [12] and [13].

From a specific first order partition, the Linear programming relaxation is used to identify a set of promising variables. This set is used to implement a second order partition. The overall goal of partitioning is to solve a number of easier problems, rather than one difficult problem, where the solution times of the easier problems should be relatively short.

An algorithm based on these ideas was developed for the TKP algorithm and is published in an operations research journal, [19] and presented at various conferences by the authors, [18]. The algorithm was ported to the ETKP case as well. The algorithm is presented in the appendix for the interested reader and for the sake of completeness.

The ETKP partitioning algorithm initially produced poor computation results. This was due to a large number of second order partitions to investigate. To enhance the computational efficiency several additional ideas were implemented.

The first such idea is to add valid inequalities to the formulation. The idea of valid inequalities is discussed in more detail in [9] and [10]. A large integrality gap (the difference between the best integer solution found so far and the linear programming relaxation) leads to too many second order partitions to investigate. Adding valid inequalities reduced the integrality gap and limited the number of second order partitions to investigate, improving solution times dramatically. The valid inequalities used are also presented in the Appendix to this paper.

A heuristic was also developed to obtain good starting solutions. Another use of the heuristic is to differentiate between first order partitions. The idea of the heuristic is to identify a set of promising nodes in the sense that the solution to the LP relaxation included these nodes by assigning values of 1 in the optimal LP solution although the variables were not constrained to be zero or one. A simple 0-1 knapsack is formed with nodes adjacent to the promising set. This problem is relatively easy to solve. This heuristic gave good bounds and in some cases produced the optimal solution after one application and was subsequently proved by the application of the branch and bound process.. The heuristic is also given in the Appendix.

VI. RESULTS

Computational experiments were done using CPLEX from ILOG as the standard off the shelf optimization software. This software was used to solve all linear-, mixed integer linear- and integer programming problems. This also presented a bench mark for the enhanced modeling and

partitioning algorithm. Since numerous errors were found in the algorithm presented in [15] this algorithm was not developed and could not be tested for computational efficiency.

Problems were generated using a pseudo random number generator. Problems sized ranged between 500 and 10,000. For all problems a time cut-off value of 7,200 seconds was imposed. Because the ETKP may need to be solved multiple times as it forms part of the modelling of larger telecommunications networks, it was deemed reasonable to impose a cut-off value of 7,200 seconds. If a user employs these methods to solve a particular design problem, these limits can obviously be extended. For each problem size 4 different tree configurations were generated. Within each configuration 9 different capacities were used. This is in contrast to work presented in [15] where the capacities of the problem instances were kept fixed even when the problem size was increased. It was felt that this practice may lead to a misrepresentation of the actual performance of the algorithm published in [15].

Without the added valid inequalities the partitioning algorithm did not produce viable solution times to problems sizes above 300 nodes. In the empirical work reported on here the partitioning method with valid inequalities and the heuristic was employed. This is due to the fact that very few data instances could be solved with more than 300 nodes without the valid inequalities, as there were too many second order partitions to investigate and too large an integrality gap.

For the empirical work AMD DL145G2 OPTERON 64 bit machines were used. The computers form part of a cluster computing platform with parallel computing capabilities. Parallel computing was not utilised during the computational test, but may be considered in future work. The operating system is RedHat Linux. The machines have 4Gb of ram and CPLEX 10 was used.

The legend for the graph and data presented is as follows:

- CPLEX: Standard optimization software from ILOG, considered to be industry standard software.
- Part ETKP Alg: The partitioning algorithm

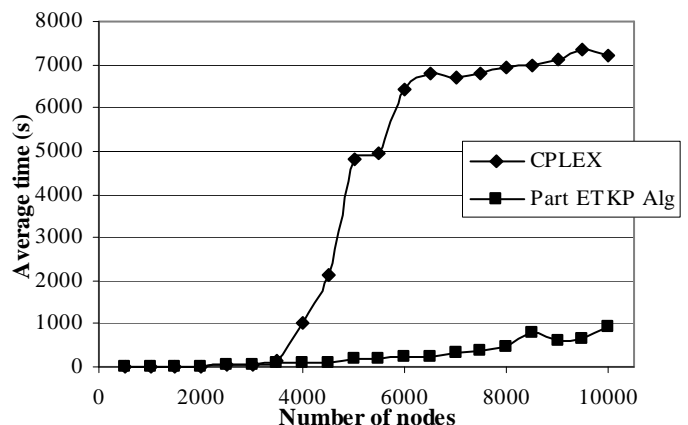


Fig 1. Average solution times for ETKP data instances

presented implementing enhanced modeling and partitioning strategies.

In Table I the standard deviation is presented for the

TABLE I
STANDARD DEVIATION OF CPLEX AND PARTITIONING
ALGORITHM

Nodes	CPLEX	Part ETKP Alg
500	1.104	0.861
1000	5.157	2.344
1500	7.073	6.788
2000	10.210	10.614
2500	21.585	17.217
3000	72.149	23.899
3500	140.059	27.254
4000	1773.873	45.559
4500	2713.800	59.641
5000	2845.631	135.717
5500	3038.496	95.180
6000	2307.930	119.173
6500	1861.777	129.568
7000	1987.087	244.594
7500	1849.912	238.260
8000	1643.248	318.430
8500	1342.197	1174.457
9000	896.310	352.911
9500	23.484	374.184
10000	796.818	752.070

various data instances

It can be seen from the graph and the data that on average the partitioning algorithm solved the ETKP data instances in less time than standard CPLEX. The solutions obtained are exact proven optimal solutions. It is evident from the graph that the average CPLEX is close to the cut-off value of 7,200 seconds for larger problem instances, meaning that CPLEX could not reliably solve the data instances.

Generally the standard deviations for the partitioning ETKP algorithm are also less than the deviations of CPLEX, see Table I above. This means that the solution times were less variable for the partitioning algorithm than for CPLEX. The exception is for some of the larger problem sizes. This is due to fact that in many cases CPLEX reached the cut-off value, which means there is little variation in solution times, but reaching the cut-off time is not a desired outcome, since the procedure was stopped before the optimal solution was obtained.

VII. CONCLUSION

Solving ETKP problems with a partitioning algorithm that uses standard off the shelf software coupled with enhanced modeling seems to be a very viable solution strategy. This can be very helpful as the ETKP is a basic subproblem in LATN design problems. The main conclusion is that the empirical experience with the new algorithms indicate that problems with more than 6 000 nodes can be reliably solved whereas the other methods tested failed to do so in reasonable time.

VIII. FUTURE WORK

Future work may be to investigate different cost functions and to see if the approach of using standard software coupled with enhanced modeling and partitioning can be

used in other models for LATN design.

APPENDIX

The partitioning algorithm implemented and enhancements are presented in this section.

1) Partitioning ETKP algorithm

Define ILP(ETKP) as an Integer Linear Programming model for the ETKP with ILPR(ETKP) the Linear Programming relaxation of the problem. In the same way ILP(ETKP, p) defines a first order MILP partitioning with corresponding relaxation being ILPR(ETKP, p). The second order partition is indicated ILP(ETKP, p,q) for the MILP and ILPR(ETKP, p,q) for the relaxation. For a full definition of the partition, refer to [17]. The partitions are also given implicitly in the procedure below.

procedure PART_ETKP

begin

Set $CLB = -\infty$

Set $Z_{ETKPR} =$ Solution to ILPR(ETKP) and denote the solution values by x_j^* for $j = 0, 1, 2, \dots, n-1$

Define $P = \{1, 2, 3, \dots, n-1\}$

Solve parametrically ILPR(ETKP, p) for $p \in P$ to obtain associated objective function values $Z_{ETKPR(p)}$. If

ILPR(ETKP, p) is infeasible, set $Z_{ETKPR(p)} = -\infty$.

while $P \neq \emptyset$ **do**

begin

Set

$p' = t$ where t is defined by $Z_{ETKPR(t)} = \max_k \{Z_{ETKPR(k)} \mid k \in P\}$

For

$Z_{ETKPR(p')}$ identify the solution values x_j' for $j = 0, 1, 2, \dots, n-1$

Define $S_l = \{j \mid x_j' = 1, j = 0, 1, 2, \dots, n-1\}$ and set

$S_{n-l} = V \setminus S_l$

Set $l = \sum_{j \in S_l} x_j'$

Set $Q = \{l, l-1, l-2, \dots, \max\{0, p - |S_{n-l}|\}\}$

Solve parametrically ILPR(ETKP, p',q) for $q \in Q$ to obtain associated objective function values of ILPR(ETKP, p',q) denoted by $Z_{ETKPR(p',q)}$, if

ILPR(ETKP, p',q) is infeasible set $Z_{ETKPR(p',q)} = -\infty$

while $Q \neq \emptyset$ **do**

begin

Set $q' = s$ where s is defined by

$Z_{ETKPR(p',s)} = \max_j \{Z_{ETKPR(p',j)} \mid j \in Q\}$

if $Z_{ETKPR(p',q')} > CLB$ **then**

begin

Solve ILP(ETKP, p',q') with corresponding

solution values $x_j^{\wedge}, j = 0, 1, 2, \dots, n-1$ and variables

$\delta_i^{\wedge}, i = 1, 2, \dots, n-1$

Set $Z_{ETKP(p',q')} =$ Solution to ILP(ETKP, p',q')

if $Z_{ETKP(p',q')} > CLB$ **then**

```

begin
  Set  $CLB = Z_{ETKP(p',q)}$ 
  Update  $x_j^{CLB} = \hat{x}_j, j = 0, 1, 2, \dots, n-1$ 
  Set  $Q = \{r \mid r \in Q \text{ and } Z_{ETKPR(p',r)} > CLB\}$ 
end
end
  Set  $Q = Q \setminus \{q\}$ 
end.
  Set  $P = P \setminus \{p\}$ 
  Set  $P = \{i \mid i \in P \text{ and } z_{ETKPR(i)} > CLB\}$ 
end (while)
Optimal solution = CLB
end.

```

2) Valid inequalities

The first valid inequality tries to exploit the fact that capacity is used up on a path between a node and the gateway or root node. This, in combination with contiguity constraints, force expansion higher up in the network if a node is added.

procedure add_cut1

```

begin
for  $i = 1$  to  $n-1$  do
begin
   $j = i$ 
  sub_cap = 0.0
do
    sub_cap = sub_cap +  $d_j$ 
    if sub_cap  $\geq b_j$ 
then add_constraint  $x_i \leq \delta_j$ 
     $j = p_j$ 
while  $j \neq \text{root\_node}$ 
end
End.

```

The second valid inequality aims to exploit the capacity required when adding all the successor nodes of a specific node. If the sum of the capacities of the successor nodes of node i exceeds the capacity b_i available at node i , expansion would be required before all the successor nodes can be added.

procedure add_cut2

```

begin
for  $i = 0, 1, 2, \dots, n-1$  do
begin
  Define  $C_i = \{j \mid j \text{ child node of } i\}$  and
   $|C_i| = \text{number of child nodes of } i$ 
  if  $C_i \neq \emptyset$  then
begin
    dem_sum =  $\sum_{k \in C_i} d_k$ 
I if dem_sum  $\geq b_i$  then do
begin
      add_constraint  $x_i + \sum_{j \in C} x_j - \delta_i \leq |C_i|$ 
end
end

```

end

Before the next valid inequality is discussed it is first necessary to define further notation. Let $P(i, j)$ define the set of nodes on the path between node i and node j . With this notation node i is not included in the set.

procedure add_cut3

```

Begin
for  $i = 1$  to  $n-1$  do
begin
  add_constraint  $y_{i2} \leq \left( H - \sum_{j \in P(0,i]} d_j - b_i \right) \delta_i$ 
end
End.

```

This valid inequality uses the capacity used up on the path between the root node and the node for which the inequality is added and limits the additional flow y_{i2} .

Another enhancement is due to the fact that the ETKP capacity H in the equation $y_{i2} \leq H \delta_i$ enables the LP relaxation to obtain a solution that is too optimistic and contributes to the large integrality gap. It was found that the following valid type of valid inequality equation $y_{i2} \leq \left(\sum_{j \in T(i)} d_j \right) \delta_i$ where

$T(j) = \{i \mid i \text{ is a descendant of } j\} \cup \{j\}$ also decrease the integrality gap.

Note that the third and fourth sets of inequalities imply that H in the equations $y_{j2} \leq H \delta_j, j = 1, 2, 3, \dots, n-1$ can be replaced by $y_{j2} \leq H' \delta_j, j = 1, 2, 3, \dots, n-1$ where $H' = \min \left\{ H; H - \sum_{k \in P(0,j]} d_k - b_j; \sum_{i \in T(j)} d_i \right\}$.

3) Heuristic method

An heuristic method was implemented after the LP relaxation of the ETKP, ILPR(ETKP), has been solved to obtain solution values x_j^* for $j = 0, 1, 2, \dots, n-1$. The set B_i is defined as the set of nodes included (with x_j values of 1) in the solution of ILPR(ETKP). Define the set B_{Add} as the set of child nodes of the set B_i . The capacity used up for the sub tree B_i is defined as $H_{B_i} = \sum_{j \in B_i} d_j$ with the capacity available for the heuristic as $H - H_{B_i}$. The ILP heuristic ILP(ETKP, H_{B_i}) is then defined as follows:

$$\begin{aligned}
 & \max \sum_{j=0}^{n-1} c_j x_j - \sum_{j=1}^{n-1} (F_j \delta_j + \alpha_j y_{j2}) \\
 & \text{s.t. } x_j - x_{p_j} \leq 0 \quad j = 1, 2, \dots, n-1 \\
 & x_0 = 1 \\
 & D\mathbf{x} - B(\mathbf{y}_1 + \mathbf{y}_2) = \mathbf{0}, \\
 & d^T \mathbf{x} \leq H, \\
 & y_{j2} \leq H \delta_j, \quad j = 1, 2, 3, \dots, n-1 \\
 & 0 \leq y_{j1} \leq b_j \quad j = 1, 2, \dots, n-1 \\
 & \sum_{j \in B_i} x_j = |B_i| \\
 & \sum_{j \in B_{Adj}} x_j d_j \leq H - H_{B_i} \\
 & y_{j1}, y_{j2} \geq 0 \quad j = 1, 2, \dots, n-1,
 \end{aligned}$$

$$x_j \in \{0,1\} \quad j = 0,1,\dots,n-1,$$

$$\delta_j \in \{0,1\} \quad j = 1,2,3,\dots,n-1.$$

ACKNOWLEDGMENT

The authors would like to thank the anonymous referees for their input towards improving the article.

REFERENCES

- [1] A. Balakrishnan, T.L. Magnanti, A. Shulman and R.T. Wong, "Models for planning capacity expansion in local access telecommunication networks," *Annals Of Operations Research*, vol. 33, pp. 239–284, 1991.
- [2] A. Balakrishnan, T. Magnanti and R.T. Wong, "A decomposition algorithm for local access telecommunications network expansion planning," *Operations Research*, vol 43, no 1, pp 58–76, 1995.
- [3] G. Cho and Shaw D.X., "A depth-first dynamic programming algorithm for the tree knapsack problem," *INFORMS Journal On Computing*, vol 9, pp 431–438, 1997.
- [4] G. Cho, Shaw, D.X. and S. Kim, "An efficient algorithm for a capacitated subtree of a tree problem in local access telecommunications networks," *Computers And Operations Research*, vol 8, pp 737–748, 1997.
- [5] M. Corte-real and L. Gouveia, "Network flow models for the local access network expansion problem," *Computers And Operations Research*, vol 34, pp 1141–1157, 2007.
- [6] O.E. Flippo, A.W.J. Kolen, A.M.C.A. Koster and R.L.M.J. van de Leensel, "A dynamic programming algorithm for the local access telecommunication network expansion problem," *European Journal Of Operational Research*, vol 127, pp 189–202, 2000.
- [7] I. Godor and G. Magyor, "Cost-optimal topology planning of hierarchical access networks," *Computers And Operations Research*, vol 32, pp 59–86 2005.
- [8] D.S. Johnson and K.A. Niemi, "On knapsacks, partitions and a new dynamic programming technique for trees," *Mathematics Of Operations Research*, vol 8, pp 1–14, 1983.
- [9] E.L. Johnson and G.L. Nemhauser, "Recent developments and future directions in mathematical programming", *IBM Systems Journal*, vol 33, no 1, pp 79–93, 1992.
- [10] J.T. Linderoth and M.W. Savelsbergh, "A computational study of search strategies for mixed integer programming," *INFORMS Journal On Computing*, vol 11, no2, pp 173–187, 1999.
- [11] S. Martello and P. Toth, "Algorithms for knapsack problems," *Annals Of Discrete Mathematics*, vol 31, pp 213–258, 1987.
- [12] S. Martello and P. Toth, "Upper bounds and algorithms for hard 0-1 knapsack problems," *Operations Research*, vol 45, pp 768–778, 1997.
- [13] D. Pisinger, "Algorithms for knapsack problems," Ph.D. dissertation, University of Copenhagen, 1995.
- [14] D.X. Shaw and G. Cho, "The critical-item, upper bounds, and a branch-and bound algorithm for the tree knapsack problem," *Networks*, vol 31, pp 205–216. 1996.
- [15] D.X. Shaw, G. Cho and H. Chang, "A depth-first dynamic programming procedure for the extended tree knapsack problem in local access network design," *Telecommunication Systems*, vol 7, pp 29–43, 1997.
- [16] H.D. Sherali, Y. Lee and T. Park, "New modeling approaches for the design of local access transport networks," *European Journal Of Operational Research*, vol 127, pp 94–108, 2000.
- [17] D.J. van der Merwe, "The use of partitioning strategies in local access telecommunication network problems and other applications," Ph.D. Thesis, Dept. of Computer Science and Information Systems, North-west University, Potchefstroom Campus, 2007.
- [18] D.J. van der Merwe and J.M. Hattingh, "The feasibility of LATN design using tree knapsack and extended tree knapsack models," in 2005 *Proc SATNAC*.
- [19] D.J. van der Merwe and J.M. Hattingh, "Tree knapsack approaches for local access network design," *European Journal Of Operational Research*, vol 174, pp 1968–1978, 2006.

David J. van der Merwe holds a B.Sc in Computer Science and Mathematics, a B.Hons, M.Sc. and Ph.D. in Computer Science.

He is currently employed as a lecturer at the North-West University, Potchefstroom Campus in the School of Computer, Statistical and Mathematical Sciences.