Computational Comparison of Algorithms for a Generalization of the Node-Weighted Steiner Tree and Forest Problems

Raul Brás and J. Orestes Cerdeira

Abstract Habitat fragmentation is a serious threat for the sustainability of species. Thus, the identification of effective linkages to connect valuable ecological units is an important issue in conservation biology. The design of effective linkages should take into account that areas which are adequately permeable for some species' dispersal may act as obstructions for other species. The determination of minimum cost effective linkages is a generalization of both node-weighted Steiner tree and node-weighted Steiner forest problems. We compare the performance of different procedures for this problem using large real and simulated instances.

1 Introduction

In conservation biology, habitat fragmentation is considered a key driver of biodiversity loss [4, 10]. To mitigate the impacts of fragmentation on biodiversity, connectivity between otherwise isolated populations should be promoted [15]. To effectively promote connectivity, there is need for procedures to identify linkages (i.e., areas to establish the connection) between habitats of each of several species (i) that take into account that linkage areas for a species might be barriers for others, and (ii) that are cost-efficient, since placing linkage areas under conservation compete with other land uses.

The problem can be formulated as follows. Consider a graph $G = (V, E)$ where the nodes of V identify the cells (usually grid squares) in which the study region has been divided, and which are considered suitable for conservation actions. The edges

R. Brás (✉)
Instituto Superior de Economia e Gestão and Centro de Matemática Aplicada à Previsão e Decisão Económica, Universidade de Lisboa, Portugal
e-mail: rbras@iseg.ulisboa.pt

J.O. Cerdeira
Departamento de Matemática & Centro de Matemática e Aplicações, Faculdade de Ciências e Tecnologia, Universidade Nova de Lisboa, Lisboa, Portugal
e-mail: jo.cerdeira@fct.unl.pt

J.P. Almeida et al. (eds.), *Operational Research*, CIM Series in Mathematical Sciences 4, DOI 10.1007/978-3-319-20328-7_5

of E define adjacencies between pairs of cells (usually two cells are adjacent if they have a common border).

For each species (or group of "similar" species, i.e., sharing the same habitats and suitable areas to disperse) k, $k = 1, \dots, m$, let T^k be the set of nodes representing the habitats of species k (*terminals* of type k), and V^k the set of nodes corresponding to cells which can be used as linkage passages for species k. We assume that $T^k \subseteq V^k$, and call k-barriers to the cells of $V \setminus V^k$.

A feasible solution of the problem is a subset of nodes $S \subseteq V$ that, for $k = 1, \dots, m$, includes a path that only uses nodes of V^k between every pair of nodes in T^k.

Suppose there is a (non negative) cost associated to every node, quantifying the charge of allocating the corresponding cell to conservation purposes. The problem, which we will call multi-type linkage problem (MTLinkP for short), seeks for a minimum cost feasible solution (i.e., which minimizes the sum of the costs of the nodes).

MTLinkP, that was independently considered by Lai et al. [13] and by Alagador et al. [1], is a generalization of the node-weighted Steiner tree [20] and of the node-weighted Steiner forest [8] problems. If $V = V^k$ and $m = 1$ (i.e., only one species, no barriers) MTLinkP is the node-weighted Steiner tree problem. If $V = V^k$, for $k = 1, \cdots, m > 1$ (i.e., different species, no barriers) MTLinkP is the node-weighted Steiner forest problem.

Lai et al. [13] and Alagador et al. [1] proposed a heuristic procedure for MTLinkP by solving a sequence of node-weighted Steiner tree problems, one for each type k, and outcome the union of these m Steiner solutions. We call this approach *type by type*. In the same paper Lai et al. [13] presented a heuristic for MTLinkP that is a generalization of the primal-dual algorithm of Demaine et al. [5] for the node-weighted Steiner forest problem, and report computational results on synthetic instances of small size (up to 15×15 grids) and m up to 5, and a real instance consisting of two species, 4514 cells and up to 17 terminals.

Brás et al. [2] developed a computer application, MulTyLink, implementing a version of the type by type algorithm and a GRASP type heuristic. This software was announced in Brás et al. [3], along with a brief description of the two algorithms used by the program.

In this paper we compare the performances of the primal-dual algorithm of Lai et al. [13] and the two heuristics in MulTyLink on real and simulated data sets. We start giving in Sect. 2 a multiflow formulation for MTLinkP. In Sect. 3 we give some details on the two heuristics of MulTyLink, and summarize the primal-dual heuristic of Lai et al. [13]. In Sect. 4 we report results of computational experiments to compare running times and quality of the solutions produced with the three heuristics. We finish with some remarks in Sect. 5.

2 Multiflow Formulation

Flow formulations have been used to model Steiner tree problems (see, e.g., Wong [21] for the standard edge-weighted Steiner tree problem; Segev [18] for a special case of non negative costs on the edges and negative costs on the nodes and Magnanti and Raghavan [14] for general network design problems with connectivity requirements including the edge-weighted Steiner forest problem).

Here we give a multi-commodity flow based formulation of MTLinkP.

Let $w_v \geq 0$ be a cost associated to each node v of graph G and let $w_v = 0$ if $v \in T^k$, for some $k = 1, \cdots, m$. Denote by A the set of arcs obtained by assigning two arcs of opposite directions to every edge of G.

For $k = 1, 2 \ldots, m$, let $t_1^k, t_2^k, \cdots, t_{p_k}^k$, with $p_k = \left|T^k\right|$, be the terminals of T^k taken by some arbitrary order. Node t_1^k will supply every other node in T^k (the demanding nodes) with one unit of commodity. Variables $f_{(u,v)}^{ki}$ on arcs indicate the amount of commodity k (amount of flow of type k) along arc (u, v) with origin t_1^k and destination t_i^k, $i = 2, \ldots, p_k$. Connectivity of the nodes of T^k in solutions is ensured by the mass balance constraints which state that the amount of commodity k out of a node v minus the commodity k into v must equal the supply/demand amount.

In addition to flow variables $f_{(u,v)}^{ki}$, binary variables x_v on nodes will be used to indicate whether node v is included ($x_v = 1$) or not ($x_v = 0$) in the solution. With these variables MTLinkP can be formulated as follows.

$$\min \sum_{v \in V} w_v x_v \tag{1}$$

subject to:

$$\sum_{\{v \in V^k : (t_1^k, v) \in A\}} f_{(t_1^k, v)}^{ki} = 1, \qquad \begin{array}{l} k = 1, \cdots, m, \\ i = 2, \cdots, p_k \end{array} \tag{2}$$

$$\sum_{\{u \in V^k : (u,v) \in A\}} f_{(u,v)}^{ki} = 1, \qquad \begin{array}{l} v \in T^k \setminus \{t_1^k\} \\ k = 1, \cdots, m, \\ i = 2, \cdots, p_k \end{array} \tag{3}$$

$$\sum_{\{u \in V^k : (v,u) \in A\}} f_{(v,u)}^{ki} - \sum_{\{u \in V^k : (u,v) \in A\}} f_{(u,v)}^{ki} = 0, \qquad \begin{array}{l} v \in V^k \setminus \{T^k\} \\ k = 1, \cdots, m, \\ i = 2, \cdots, p_k \end{array} \tag{4}$$

$$\sum_{\{u \in V^k : (u,v) \in A\}} f_{(u,v)}^{ki} \leq x_v, \qquad \begin{array}{l} v \in V^k \setminus \{t_1^k\}, \\ k = 1, \ldots, m, \\ i = 2, \cdots, p_k \end{array} \tag{5}$$

$$x_v \in \{0, 1\}, \qquad v \in V \tag{6}$$

$$f_{(u,v)}^{ki} \geq 0, \quad u, v \in V^k, \quad (u, v) \in A, \quad k = 1, \ldots, m, \quad i = 2, \cdots, p_k. \tag{7}$$

The mass balance equations (2), (3) and (4) dictate that one unit of flow of type k will be routed between the supply node t_1^k and each demanding node of $T^k \setminus \{t_1^k\}$. The "capacity" constraints (5) ensure there is no flow along the arcs entering nodes that are not included in the solution. Constraints (2), (3), (4), (5), (6) and (7) guarantee the existence of a directed path between t_1^k and every other node of T^k, thus ensuring that all nodes of T^k will be in the same connected component of the solution.

We will use the compact formulation (1), (2), (3), (4), (5), (6) and (7) above to derive, from a mixed integer programming solver, MTLinkP optimal values for small size instances, to assess the quality of the solutions produced by the heuristic algorithms of Sect. 3.

3 Heuristics

3.1 Type by Type Heuristic

Given a permutation $i_1, i_2, \ldots, i_m$ of (integer) types $1, 2, \ldots m$, the *type by type* heuristic computes in step k a Steiner solution with respect to $< V^{i_k} >$, the subgraph of G induced by V^{i_k}, updates the costs of nodes v of that solution letting $w_v = 0$, and proceeds to the next step $k + 1$. The final MTLinkP solution results from turning minimal feasible (with respect to inclusion) the union of nodes of the Steiner solutions obtained in each step.

The process can be repeated for a number of different permutations of integers $1, 2, \ldots, m$, and the best solution is returned (see Fig. 1).

To solve the node-weighted Steiner tree problem in each step, we use the following straightforward modification of the well known *distance network heuristic* suggested by Kou et al. [12] for the edge-weighted Steiner tree. If H is a graph

```
1. $Sol \leftarrow \emptyset$
2. $w(Sol) \leftarrow \infty$
3. $\mathscr{P} \leftarrow$ subset of permutations of $\{1,\ldots,m\}$.
4. For all $P \in \mathscr{P}$
   a. $X \leftarrow \emptyset$
   b. For all $k \in P$
      i. Build graph $< V^k >$ with weights: 0 if $v \in X$, $w_v$ otherwise.
      ii. $X^k \leftarrow$ Steiner solution w.r.t. $< V^k >$ and $T^k$
      iii. $X \leftarrow X \cup X^k$
   c. Turn $X$ minimal. For each $v \in X \setminus \bigcup_k T^k$ (randomly ordered) remove $v$ from $X$ if
      $X \setminus \{v\}$ is MTLinkP feasible.
   d. $w(X) \leftarrow \sum_{v \in X} w_v$
   e. If $w(X) < w(Sol)$ then
      i. $Sol \leftarrow X$
      ii. $w(Sol) \leftarrow w(X)$
5. Return $Sol$.
```

Fig. 1 TbT heuristic

with costs w on the nodes and terminal set S, we define the distance network $D(S)$ which is the complete graph with S as its node set, and where the weight of every edge (u, v) is the cost of the minimum cost path connecting terminal u to v on H. Note that determining a node-weighted shortest path on undirected graph H between nodes u and v, with $w_u = w_v$, reduces to finding an edge-weighted shortest path from u to v in the digraph obtained assigning opposite directions to every edge of H, and defining the cost of every arc (i, j) as being equal to w_j.

A minimum spanning tree of $D(S)$ is determined and a (node-weighted) Steiner solution N is defined as the set of the nodes of the shortest paths corresponding to the edges of the spanning tree.

In the final step the nodes of N are considered randomly and node j is removed from N if all nodes of S belong to the same connected component of the subgraph of H induced by $N \setminus \{j\}$.

We use the above modification of the *distance network heuristic* since it is fast and does not use large data structures. Procedures such as Klein and Ravi [11] heuristic, based on the Rayward-Smith [16] algorithm, that perform well for node-weighted Steiner problems, would be impractical for the large size instances of MTLinkP we want to handle. Klein and Ravi [11] heuristic needs to compute the minimum cost paths between all pairs of nodes, which is time consuming and requires large amounts of memory.

Lai et al. [13] version of this heuristic uses the Dreyfus-Wagner (DW) algorithm [7] to solve the Steiner problem at each step. DW is an exact dynamic programming algorithm that runs in exponential time, not suitable to solve the instances that we present in this paper.

3.2 Primal Dual Heuristic

Lai et al. [13] gave a modified version of the Demaine et al. [5] heuristic for node-weighted Steiner forest problems. The heuristic operates on the following cut-covering formulation of MTLinkP. Minimize (1) subject to (6), and

$$\sum_{v \in \Gamma^k(S)} x_v \geq f^k(S), \qquad \begin{array}{l} S \subseteq V^k \\ k = 1, \ldots, m \end{array} \tag{8}$$

where $f^k(S) = 1$, if $\emptyset \neq S \cap T^k \neq T^k$ (i.e., if S includes at least one terminal of T^k, but not all), and $f^k(S) = 0$, otherwise, and $\Gamma^k(S)$ is the set of nodes $v \in V^k \setminus S$ adjacent to at least one node in S.

The dual of the linear relaxation of (1), (6), (8) is:

$$\max \sum_{k=1}^{m} \sum_{S \subseteq V^k} f^k(S) y^k(S)$$

```
1. $X \leftarrow \bigcup_k T^k$
2. For $k = 1,\dots,m$ calculate $\mathscr{C}(\langle X^k \rangle)$. $y^k(C) \leftarrow 0$ for every $C \in \mathscr{C}(\langle X^k \rangle)$
3. While $X$ is not MTLinkP feasible
    a. Simultaneously increase $y^k(C)$ until, for some $v$, $\sum_{k=1}^m \sum_{C \subseteq V^k : v \in \Gamma^k(C)} y^k(C) = w_v$.
    b. $X \leftarrow X \cup \{v\}$
    c. For $k = 1,\dots,m$ recalculate $X^k$ and $\mathscr{C}(\langle X^k \rangle)$.
4. Turn $X$ minimal. For each $v \in X \setminus \bigcup_k T^k$ (by reverse order of insertion) remove $v$ from $X$
   if $X \setminus \{v\}$ is feasible
5. Return $X$.
```

Fig. 2 PD heuristic

subject to:

$$\sum_{k=1}^{m} \sum_{S \subseteq V^k : v \in \Gamma^k(S)} y^k(S) \leq w_v \quad v \in V$$

$$y^k(S) \geq 0 \qquad \begin{matrix} S \subseteq V^k \\ k = 1,\dots,m \end{matrix}$$

The heuristic maintains an infeasible primal solution X, and dual variables $y^k(S)$. The algorithm is described in Fig. 2, where $\mathscr{C}(\langle X^k \rangle)$ are the connected components of the graph induced by $X^k = X \cap V^k$.

3.3 GRASP Heuristic

The type by type (TbT) heuristic and the primal-dual heuristic (PD) of Lai et al. [13] define a feasible solution adding in each step nodes to a current unfeasible solution X. TbT heuristic adds to X a set of nodes that guarantee the connection of all terminals from a certain predetermined type, and assigns costs equal to zero to all the added nodes. PD heuristic adds to X one single node that belongs to V^k and is adjacent to X^k, for at least one not previously determined type k. We present a kind of greedy randomized adaptive search procedure (GRASP) [9] that hybridizes the two heuristics.

GRASP starts with set X consisting of all terminals of T^k, $k = 1 \cdots , m$, and in each step grows the current unfeasible solution X as follows. First, some type k, for which not all terminals of T^k are connected, is uniformly selected. Next, a connected component S of $< X^k >$, the subgraph induced by X^k, that includes terminals of type k, is uniformly selected, and a minimum cost path P, among the minimum cost paths connecting S with every other component of $< X^k >$ that includes terminals of type k, is determined. The nodes of P are added to X, and costs are updated letting $w_v = 0$ to every node v of P. Note that, since the costs of nodes of X are all equal to zero,

```
1. $w(Sol) \leftarrow \infty$
2. $r \leftarrow$ number of repetitions
3. For $i = 1, \ldots, r$
   a. $X \leftarrow \bigcup_k T^k$
   b. While $X$ is not MTLinkP feasible
      i. $X^k = X \cap V^k$. Calculate $\mathscr{C}(\langle X^k \rangle),\ k = 1, \ldots, m$
      ii. $Q = \{k$ : not all terminals of $T^k$ belong to the same $S \in \mathscr{C}(\langle X^k \rangle),\ k = 1, \ldots, m\}$
      iii. If $Q = \emptyset$ then end the while cycle
      iv. $p \leftarrow$ member of $Q$ uniformly selected
      v. $S \leftarrow$ member of $\mathscr{C}(\langle X^p \rangle) : S \cap T^p \neq \emptyset$ uniformly selected
      vi. $P \leftarrow$ minimum cost path connecting $S$ to $U \in \mathscr{C}(\langle X^p \rangle) \setminus S : U \cap T^p \neq /\ 0$
      vii. $X \leftarrow X \cup P,\ w_v = 0,\ \forall v \in P$
   c. Turn $X$ minimal. For each $v \in X \setminus \bigcup_k T^k$ (randomly ordered) remove $v$ from $X$ if
      $X \setminus \{v\}$ is MTLinkP feasible.
   d. $w(X) \leftarrow \sum_{v \in X} w_v$
   e. If $w(X) < w(Sol)$ then
      i. $Sol \leftarrow X$
      ii. $w(Sol) \leftarrow w(X)$
4. Return $Sol$.
```

Fig. 3 GRASP heuristic

P can be easily obtained with Dijkstra algorithm [6], choosing an arbitrary node in S as the starting node and ending as soon as a node of a component of X^k, including terminals of T^k and different from S, is added to the path.

The final GRASP solution is obtained by turning minimal feasible the solution X obtained in the last step. Given its random behavior, repeating GRASP a number of times with the same input is likely to produce different solutions, and the best solution is outcome (see Fig. 3).

4 Computational Experiments

We performed computational tests to evaluate the quality of the solutions produced by the heuristics, as well as the practicality of the flow formulation of Sect. 2.

4.1 General Case

Here we report results for the case where not all V^k coincide.

4.1.1 Data

We used real and simulated instances to test the heuristics.

1. Real Data.

Real data concerns the linkage of climatically-similar protected areas (PA) in the Iberian Peninsula (IP). IP is represented as 580,696 1 km $\times$ 1 km cells, from which 80,871 cells intersecting the 681 existent PA in the IP were defined as terminals. Terminals were clustered in four groups sharing similar climates (with respect to four climatic variables which are considered important drivers of species' distributions). Adjacency was considered in terms of common edges or corners of the square cells.

Cells with considerable human intervention (values derived from Human Footprint Index data available from http://www.ciesin.columbia.edu/wild_areas greater than 60 in a range from 0 to 100) were excluded as they were considered poorly permeable to species' movements. This has reduced the number of cells to 438,948 (which includes every protected cell).

Figure 4 (page 74) shows the location of protected cells from each class (colored cells), and cells that were excluded due to presenting high levels of human intervention (grey cells).

For $k = 1, \ldots, 4$, V^k was defined as the set of cells that do not significantly differ from the mean climatic conditions of PA of class k. This was delineated as follows. The centroid, in the climatic space, of the PA cells of each climatic class was defined, and the Euclidean distances from the climate conditions of each cell

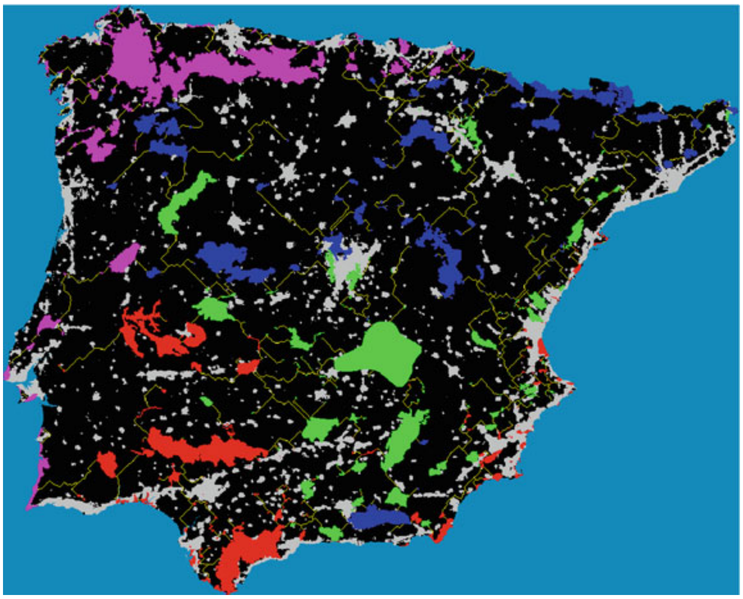

Fig. 4 Iberian Peninsula data (scenario 2). Protected cells are colored *red*, *green*, *blue* and *magenta*. *Grey* areas represent cells not in V. Cells of the solution obtained with TbT heuristic are colored *yellow*

to the centroid of each class were computed. This retrieved four values $d^k(v)$, for each cell v, expressing the dissimilarity of cell v to every climatic class k. Cell $v \in V^k$ (i.e., v was not considered k-barrier) if $d^k(v)$ is below a certain threshold value B^k. Two scenarios were considered. In scenario 1, B^k was defined as the largest dissimilarity $d^k(v)$, among the protected cells v in every PA from class k. In scenario 2, B^k was set as the third quartile of the $d^k(v)$ values for protected cells v of class k. Cell u was included in V^k (i.e., u was not considered k-barrier) if u belongs to some PA of class k, or $d^k(u) < B^k$.

The rational for the identification of linkages between climatically-similar protected areas, free from climatic barriers, stands on the assumption advocated by Alagador et al. [1] that species with similar ecological requirements occupy the same environments. Thus, linking climatically-similar protected areas is an effective way to promote the dispersal of species, counteracting in part the negative effects of fragmentation.

A cost was assigned to every non protected cell that is proportional to the cell's fraction not covered by Natura 2000 Network (w_v = (100−percentage of Natura 2000 Network covered by v)/100). The Natura 2000 network is a European scaled conservation scheme designed to complement nationally-defined protected areas. We assigned cost equal to zero to every protected cell.

Details on the IP data can be found in Alagador et al. [1].

We denote by $IP1$ and $IP2$ the IP instances under scenarios 1 and 2, respectively.

2. Simulated Data.

Simulated data were generated as follows. Each node of V is a cell from a $n \times n$ grid. Two cells are adjacent if they have a common edge or corner.

To define V_k we start by uniformly selecting an integer $s \in [0, m]$ and assume that species $1, \dots, s$ are "specialist" (can only thrive in a narrow range of environmental conditions) and species $s+1, \dots, m$ are "generalist" (are able to thrive in a wide variety of environmental conditions). Each node v of V is included in V_k with probability $1/4$ for each "specialist" species $k \leq s$, and with probability $3/4$ for each "generalist" species $k > s$. The number of terminals of each type was obtained from a discrete uniform distribution in interval $[2, \max\{|V|/1000, 5\}]$, and terminals chosen uniformly among the nodes of V^k.

We assigned to every node in $V \setminus (\cup_k T^k)$ a cost from an uniform distribution in $[0, 1]$, and cost zero to every node of T^k.

We generated small instances with $n = 10, 20, 30, 40, 50$ and $m = 2, 3, 4, 6, \ 8, 10$ and large instances with $n = 100, 200, 300, 400, 500$ and $m = 4, 6, 8, 10$.

For the same values of n and m we generated 10 instances. This gave a total of 500 instances.

4.1.2 Results

Here we report the main results of the computational tests that we carried out.

Heuristics were implemented in C++, using the Boost Graph Library [19] to calculate spanning trees, shortest paths and connected components. Parallel programming was not used and so they ran in a single thread. All times refer to elapsed times. The computers were dedicated to running the instances, so that elapsed time is close to CPU time. Solutions for the Iberian Peninsula data were obtained with a Intel Core2 Quad CPU Q9450 @2.66 GHz and 4 GB of memory machine, while for simulated data the solutions were obtained in a machine with 2 AMD Opteron 6172 processors (24 cores) @2.1 GHz and 64 GB of memory.

1. Real Data.

Table 1 displays results obtained for the Iberian Peninsula's data with each of the three heuristics. The first column identifies the problem instance (scenarios 1 and 2). Each of the three pairs of the remaining columns contains the value of the solution obtained with a heuristic: GRASP, type by type (TbT) and primal-dual (PD), and the corresponding running time in seconds. The TbT heuristic ran for every permutation of the $m = 4$ types, while GRASP was limited to 2 hours of execution. We let the program finish the current repetition i, if it has started before the time expired, thus computation times can exceed 7200 seconds. PD was not time-limited in order to produce a solution.

GRASP obtained the best solutions. The costs of the solutions produced by TbT were slightly higher, but the times to run the 24 permutations of the four types were lower than the 2 hours that limited the execution of GRASP. PD had a poor performance. Long computation times were necessary to obtain solutions with costs that are greater than those of the solutions obtained with GRASP and with TbT. This negative behavior of PD can be explained by the specific structure of these graphs. Nodes which are far apart on the grid are connected by long paths. Thus, it is likely that PD includes a large number of redundant nodes until a feasible solution is reached. Solutions with many redundant nodes are difficult to turn minimal. The process is time consuming and produces poor solutions. GRASP and TbT, in each step, add to the solution that is being constructed the nodes of a minimum cost path connecting a pair of terminals. Thus, the number of redundant nodes is likely to be much less than that generated by PD.

Figure 4 shows a solution, obtained with the TbT heuristic, for the IP2 instance. Protected cells are colored red, green, blue and magenta and the cells of the solution

Table 1 Results for the Iberian Peninsula

	GRASP		TbT		PD	
Instance	Cost	Time	Cost	Time	Cost	Time
*IP*1	2024.67	7782.55	2035.73	5012.39	2162.11	544,490.00
*IP*2	2121.03	7525.25	2148.49	7075.90	2167.62	347,003.00

are yellow. Grey areas represent cells not in V (human footprint over 60). For a detailed interpretation of the solution, knowledge of the location of the barriers from each type would be needed.

2. Simulated Data.

The main results derived with small and large instances for simulated data are given in Tables 2 and 3, respectively. Recall that 10 instances with the same values of $|V|$ and m were considered and, therefore, each row of Tables 2 and 3 summarizes the results of 10 instances.

Table 2 Results for small instances

$\|V\|$	m	#Opts	GRASP			TbT				PD			
			% dev. from			% dev. from				% dev. from			
			Opt	BestH	Best	Opt	BestH	Best	Time	Opt	BestH	Best	Time
100	2	10	1.67	1.67	9	1.67	1.67	9	0.00	2.15	2.15	9	0.00
	3	10	1.66	1.66	9	1.66	1.66	9	0.01	0.00	0.00	10	0.00
	4	10	4.02	1.94	8	3.87	1.79	9	0.06	6.49	4.20	6	0.00
	6	10	2.32	2.17	9	3.31	3.16	8	5.19	1.61	1.48	7	0.00
	8	10	1.21	0.00	10	1.21	0.00	10	44.43	2.06	0.86	5	0.01
	10	10	0.31	0.31	9	1.27	1.27	7	60.51	0.75	0.75	8	0.01
400	2	10	0.34	0.00	10	2.51	2.18	5	0.01	7.53	7.17	5	0.02
	3	10	1.80	0.25	9	4.55	2.90	5	0.01	11.47	9.71	4	0.02
	4	10	1.60	0.00	10	4.48	2.79	5	0.06	9.02	7.27	3	0.03
	6	10	2.08	0.95	8	3.56	2.38	7	1.49	9.06	7.87	1	0.04
	8	10	0.90	0.10	9	3.70	2.87	6	18.21	9.50	8.62	2	0.06
	10	10	2.24	0.00	10	6.33	3.95	3	43.46	11.20	8.69	0	0.15
900	2	10	0.85	0.00	10	2.60	1.72	6	0.01	5.95	4.96	6	0.07
	3	10	0.54	0.00	10	1.82	1.28	8	0.02	8.15	7.50	5	0.09
	4	10	1.34	0.00	10	4.66	3.26	4	0.08	15.41	13.82	3	0.14
	6	8	2.71	0.19	9	6.54	4.55	3	3.57	19.06	15.32	2	0.27
	8	7	2.86	0.00	10	4.70	2.05	2	14.31	13.09	13.72	0	0.37
	10	7	1.82	0.39	8	3.25	2.75	5	42.45	10.26	11.07	2	0.56
1600	2	7	1.49	0.22	9	1.92	1.34	4	0.02	7.93	7.66	3	0.22
	3	5	0.09	0.00	10	0.49	4.01	4	0.05	0.65	8.51	4	0.36
	4	6	0.69	0.20	8	1.17	2.38	5	0.21	5.92	10.07	5	0.53
	6	5	1.52	0.43	8	1.93	1.13	5	4.33	4.39	16.84	3	0.74
	8	5	0.29	0.00	10	1.49	2.80	3	12.44	10.56	11.81	2	0.77
	10	1	0.00	0.06	9	1.47	4.27	1	36.13	3.32	15.09	0	1.96
2500	2	4	0.77	0.00	10	0.77	3.35	5	0.04	0.77	5.62	4	0.75
	3	6	0.00	0.08	9	0.98	1.11	6	0.07	0.98	6.91	4	0.67
	4	6	1.52	0.51	9	0.64	2.39	6	0.24	1.96	6.48	4	0.78
	6	4	5.05	0.15	8	5.57	2.41	3	6.01	4.65	7.39	4	1.71
	8	2	1.24	0.00	10	3.35	3.83	2	17.60	22.30	15.73	2	2.60
	10	0		0.04	8		1.59	6	16.20		16.19	3	2.60

Table 3 Results for large instances

$\|V\|$	m	GRASP		TbT			PD		
		%dev.	Best	%dev.	Best	Time	%dev.	Best	Time
10,000	4	0.00	10	5.27	1	1.56	18.17	1	43.71
	6	0.00	10	4.55	1	49.93	19.91	1	89.57
	8	0.00	10	3.43	2	214.82	11.90	2	67.37
	10	0.00	10	4.86	0	766.98	21.31	0	148.57
40,000	4	0.00	10	3.13	2	52.95	14.74	2	1292.83
	6	0.00	10	3.50	0	732.16	7.55	0	1116.29
	8	0.00	10	3.44	1	982.21	14.08	1	1709.82
	10	0.00	10	3.59	1	1320.94	13.60	1	1998.23
90,000	4	0.00	10	2.46	3	185.65			
	6	0.00	10	2.21	2	1056.98			
	8	0.00	10	3.66	0	1654.99			
	10	0.13	9	3.73	1	1823.81			
160,000	4	0.00	10	2.24	4	819.54			
	6	0.00	10	2.42	2	1325.13			
	8	0.03	9	2.34	2	1931.62			
	10	0.00	10	2.14	0	1947.12			
250,000	4	0.00	10	3.26	0	1378.50			
	6	0.00	10	2.45	0	1821.26			
	8	0.00	10	2.76	0	2053.90			
	10	0.00	10	2.09	2	2297.54			

We established common elapsed time limit values for the heuristics. One minute for small instances and 30 minutes for large instances, but we allowed GRASP to finish the current repetition i, and TbT to finish the current permutation P.

For most of small instances we were able to obtain optimal solutions from the flow formulation (1), (2), (3), (4), (5), (6) and (7), using CPLEX 12.4, with parallel mode set to opportunistic and 24 parallel threads (all other options used default values). For each instance, CPLEX execution time-limit was set to 1 hour of elapsed time, meaning up to 24 hours of CPU time since the machine has 24 cores.

In Table 2 column *#Opts* indicates the number of instances for which optimal solutions were found. The two columns labeled *% dev. from* indicate, for each heuristic, the mean of the relative deviations (in percentage) from the optimal values (*opt*) and from the best values of the heuristic solutions (*bestH*). The relative deviation is calculated by the expression $100(h - w^*)/w^*$, where h is the value of the heuristic solution, and w^* is the optimal value (*opt*), or the minimum of the values of the three heuristic solutions (*bestH*), respectively. The number of optimal values with respect to which averages were computed is given in column *#Opts*. Columns *best* report the number of instances for which the heuristic found the best value among the values of the three solutions obtained for the same instance with the three heuristics. Columns *time* indicate the mean computation times (in seconds) for TbT and PD. The computation times are not reported for GRASP since it uses all the amount of time allowed.

Table 3 is similar except that there are no columns regarding optimal values, since CPLEX was unable to handle the large instances. Thus, columns *% dev.* and *best* refer to comparisons with the best values of heuristic solutions.

GRASP was clearly superior for the instances considered, while PD had a poor performance.

For small instances the average over the 30 values of column *% dev. from opt* in Table 2 was 1.48 for GRASP, 2.81 for TbT and 7.11 for PD. Only four of these 30 values exceeded 2.5 % for GRASP, while six values exceeded 4.5 % for TbT. GRASP was a best heuristic in at least eight instances out of the 10 with the same $|V|$ and m. Considering all the 300 small instances, GRASP was a best heuristic in 275, TbT in 161 and PD in 116.

For the large instances the superiority of GRASP was even more evident. It has obtained the best results in 198 out of 200 instances. The mean relative deviations between TbT results and the best heuristic values were always below 5.3 %, but it attained the best result only on 24 instances.

Results on simulated data confirmed the bad behavior of PD with this kind of instances. For $|V| \geq 90{,}000$, we were unable to find solutions within the time limit of 30 minutes, except for a few instances. These were not considered in order to not bias the analysis of the results. The corresponding entries are blank on Table 3. In general, solutions were of poor quality. It seems that PD has difficulties dealing with instances where graphs have the structures here considered. An explanation was previously given when analyzing the results on the IP instances.

A fact that should be mentioned is that, several times, TbT succeeded to complete its computations within the time limits established, despite the relative high values of m ($m = 8, 10$). This is justified by the way instances were generated. Each cell of the $n \times n$ grid belongs to V^k with probability 1/4 for "specialized" species k and 3/4 for "generalist" species k. Thus, it may happen that all components of the subgraph induced by V^k, particularly for "specialized" species k, include at most one terminal of T^k, i.e., every path connecting any two terminals of T^k include some k-barrier. In this case there is no need to consider species k, as no two terminals of T^k can be linked in V^k. Since "specialist" species were uniformly chosen among the m species, the number of species that needs to be considered might be much smaller than m.

4.2 Case Where All V^k Coincide

MTLinkP is a generalization of the node-weighted Steiner tree and of the node-weighted Steiner forest problems. Therefore, GRASP, TbT and PD can be used, with no modification, to solve those problems.

We carried out some computational tests to assess how the heuristics perform on solving node-weighted Steiner forest problem.

For node-weighted forest problem, the PD heuristic is the Demaine et al. [5] algorithm. Another heuristic, based on the Rayward-Smith algorithm [16, 17] that performs well in practice for node-weighted Steiner forest is the Klein and Ravi [11] heuristic.

Klein and Ravi heuristic (KR) begins by computing the matrix M of the costs of minimum cost paths between every pair of nodes in V. Then, starting with X consisting of all terminals of T^k, $k = 1, \cdots, m$, in each step, KR adds to X the nodes of certain paths that connect a number of connected components of $< X >$, the subgraph induced by X. The connected components to merge are selected from the values of a function f that is calculated as follows, for every node $v \in V$. Let $\mathscr{S}$ be the set components of $< X >$ that, for some k, includes at least one node of T^k but not all, and let $\mathscr{S}_r$ be the family of all r sets of $\mathscr{S}$ (i.e., if $S_r \in \mathscr{S}_r$, $|S_r| = r$). For every $v \in V$ and $S_r \in \mathscr{S}_r$, let $w(v, S_r)$ be the sum of the costs of minimum cost paths connecting v with each of the r components in S_r. For every $v \in V$, define $f(v, r) = \min_{S_r} w(v, S_r) - (r-1)w_v$. The value $f(v, r)$ is the minimum cost of merging r components of $\mathscr{S}$ with r paths rooted at v. Note that the computation of $f(v, r)$ can be quickly achieved from matrix M. Finally, $f(v) = \min_{2 \le r \le |\mathscr{S}|} f(v, r)/r$, which is the minimum of the mean values of $f(v, r)$ with respect to r. In each step, KR adds to X the nodes of the paths from v which minimizes $f(v)$, while $\mathscr{S}$ is not the empty set. When there are no more components to merge, the heuristic proceeds turning solution X minimal.

We compared the performances of GRASP, TbT, Demaine et al. [5] (PD) and Klein and Ravi [11] (KR) heuristics on instances generated as above for simulated data, except that $V^k = V$, for $k = 1, \ldots, m$. We considered $n \times n$ grid graphs with $n = 50, 100, 200$ and $m = 2, 4, 6, 8, 10$ types of terminals. For each n and m two instances were generated. Table 4 reports costs and times (in seconds) on each instance. GRASP and TbT heuristics were restricted to 30 minutes of execution time. Computations were processed with the same machine that was used for the simulated data.

Results for GRASP and KR were very similar. KR obtained the best result in 56.7 % of the cases, while GRASP was the best heuristic in 40.0 % of the cases and PD in one case (3.3 %). TbT never obtained the best result. The mean relative gap between the value vH obtained by the heuristic H and the value vKR obtained with KR, given by $(vH - vKR)/vKR$, was 0.5 % for H = GRASP, 4.4 % for H = TbT and 3.5 % for H = PD. Considering only the cases for which KR performed better than heuristic H ($vKR < vH$), the mean of the relative gap was 3.1 % for H = GRASP, 5.2 % for H = TbT and 3.9 % for H = PD.

Results showed that GRASP performed better than KR in smaller instances, while in general KR outperforms GRASP for larger ones. However, the relative gap did not exceed 8.1 % (for an instance where $n = 200$ and $m = 8$). The values obtained by TbT were slightly greater than those produced by GRASP. This was more evident for larger instances ($n = 200$). PD obtains good results in the larger instances. For $n = 200$ and $m \geq 4$ it obtains better results than GRASP with relatively small times of execution. KR heuristic maintains in memory matrix M of the costs of minimum cost paths between every pair of nodes in V. For $n = 200$

Table 4 Results for Steiner forest

$\|V\|$	m	GRASP		TbT		PD		KR	
		Cost	Time	Cost	Time	Cost	Time	Cost	Time
2500	2	19.32	1800.28	20.12	0.11	20.09	1.15	19.20	2.04
		12.61	1800.37	13.01	0.08	15.92	2.34	14.67	1.92
	4	24.45	1800.18	25.31	2.64	26.36	1.57	25.04	2.68
		22.78	1800.48	23.23	2.46	25.53	1.87	22.98	2.33
	6	24.39	1800.65	24.83	104.46	25.38	0.82	24.64	2.73
		30.03	1800.47	30.74	98.04	31.56	1.47	30.44	2.85
	8	36.56	1801.00	38.66	1800.60	38.32	1.67	36.86	4.58
		29.65	1800.23	30.27	1800.37	30.50	0.81	29.97	3.57
	10	34.30	1800.63	37.12	1801.13	36.90	1.82	35.51	3.82
		33.46	1801.23	34.80	1800.23	33.51	1.01	33.30	4.71
10,000	2	46.64	1800.54	48.39	0.73	48.02	17.43	46.83	39.35
		42.19	1800.91	43.96	0.78	43.16	25.60	41.97	48.66
	4	62.06	1800.88	65.60	18.54	65.36	26.49	61.77	58.09
		47.44	1801.16	50.20	12.94	49.05	22.79	47.35	48.61
	6	78.49	1800.33	79.95	956.15	81.00	42.66	77.16	64.57
		67.07	1801.34	68.97	655.02	68.02	18.02	67.31	60.66
	8	74.07	1800.78	75.25	1801.68	75.34	17.31	71.36	72.82
		79.73	1800.39	81.12	1801.94	82.17	22.02	78.56	82.25
	10	100.02	1801.10	100.70	1802.77	99.83	59.72	98.07	96.04
		90.77	1801.33	95.66	1802.23	92.31	32.39	89.00	96.32
40,000	2	178.80	1801.51	192.27	9.94	189.42	501.73	189.39	1541.14
		83.15	1801.27	87.51	3.47	89.15	328.08	84.28	1154.73
	4	257.27	1806.26	265.71	190.69	253.92	1155.15	273.20	2280.98
		188.07	1800.93	197.61	151.75	187.33	349.34	182.06	1666.43
	6	337.71	1813.55	348.49	1814.00	327.30	882.97	317.66	4884.35
		225.93	1803.42	236.66	1806.29	228.74	829.09	218.27	2187.45
	8	316.43	1815.54	327.89	1821.25	303.15	686.42	292.69	4239.43
		305.71	1802.64	322.56	1812.95	300.84	515.96	291.56	3982.47
	10	297.26	1813.47	310.58	1813.78	290.70	885.06	283.34	3827.51
		372.83	1804.53	386.94	1817.54	360.27	597.45	345.79	6483.80

6 GB of memory are needed, and for $n = 300$ 30 GB are needed. Thus, KR heuristic could not be used for the real IP instances.

Given the above limitations of KR, GRASP and PD appear to be good options to solve large node-weighted Steiner forest problems for the type of graphs here considered.

5 Conclusions

In this paper we considered a mixed integer flow formulation and three heuristics for MTLinkP. The flow based formulation only permited to solve instances up to 2500 nodes, which is far below the size of the instances that occur in the context of conservation biology. For the specific structure of graphs of the instances that occur in conservation, GRASP seems to be a good option. Producing different solutions from different runs, on reasonable times, is relevant since, rather than a single solution, decision making needs to consider different options before proceeding negotiations with stakeholders. There are many issues (e.g., socioeconomic) involved in the analysis of conservation actions which are not easily quantifiable, thus having different alternatives to choose is an important feature.

Acknowledgements We are grateful to Maria João Martins and Diogo Alagador for discussion and assistance. Both authors were supported by the Portuguese Foundation for Science and Technology (FCT). R. Brás was funded by the project PEst-OE/EGE/UI0491/2013 and the *CEMAPRE (Centro de Matemática Aplicada à Previsão e Decisão Económica)* under the FEDER/POCI Programme. J. O. Cerdeira was funded through the projects UID/MAT/00297/2013, *CMA (Centro de Matemática e Aplicações)* and PTDC/AAC-AMB/113394/2009.

References

1. Alagador, D., Triviño, M., Cerdeira, J.O., Brás, R., Cabeza, M., Araújo, M.B.: Linking like with like: optimizing connectivity between environmentally-similar habitats. Landsc. Ecol. **27**(2), 291–301 (2012)
2. Brás, R., Cerdeira, J.O., Alagador, D., Araújo, M.B.: Multylink, version 2.0.2 (2012). (computer software http://purl.oclc.org/multylink)
3. Brás, R., Cerdeira, J.O., Alagador, D., Araújo, M.B.: Linking habitats for multiple species. Env. Model. Softw. **40**, 336–339 (2013)
4. Brooks, T.M., Mittermeier, R.A., Mittermeier, C.G., Rylands, A.B., da Fonseca, G.A.B., Konstant, W.R., Flick, P., Pilgrim, J., Oldfield, S., Magin, G., Hilton-Taylor, C.: Habitat loss and extinction in the hotspots of biodiversity. Conserv. Biol. **16**, 909–923 (2002)
5. Demaine, E., Hajiaghayi, M., Klein, P.: Node-weighted steiner tree and group steiner tree in planar graphs. In: Albers, S., et al. (eds.) Automata, Languages and Programming. Lecture Notes in Computer Science, vol. 5555, pp. 328–340. Springer, Berlin/Heidelberg (2009)
6. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**, 269–271 (1959)
7. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks **1**(3), 195–207 (1971)
8. Duin, C.W., Volgenant, A.: Some generalizations of the Steiner problem in graphs. Networks **17**(3), 353–364 (1987)
9. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. J. Glob. Optim. **6**(2), 109–133 (1995)
10. Hanski, I.: The Shrinking World: Ecological Consequences of Habitat Loss. Excellence in Ecology, vol. 14. International Ecology Institute, Oldendorf/Luhe (2005)
11. Klein, P., Ravi, R.: A nearly best-possible approximation algorithm for node-weighted Steiner trees. J. Algorithms **19**(1), 104–115 (1995)

12. Kou, L., Markowsky, G., Berman, L.: A fast algorithm for Steiner trees. Acta Inf. **15**, 141–145 (1981)
13. Lai, K.J., Gomes, C.P., Schwartz, M.K., McKelvey, K.S., Calkin, D.E., Montgomery, C.A.: The Steiner multigraph problem: wildlife corridor design for multiple species. In: Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, vol. 2, pp. 1357–1364 (2011)
14. Magnanti, T.L., Raghavan, S.: Strong formulations for network design problems with connectivity requirements. Networks **45**(2), 61–79 (2005)
15. Merriam, G.: Connectivity: a fundamental ecological characteristic of landscape pattern. In: Brandt, J., Agger, P. (eds.) Proceedings of the 1st International Seminar on Methodology in Landscape Ecological Research and Planning, pp. 5–15. Roskilde University, Denmark (1984)
16. Rayward-Smith, V.J.: The computation of nearly minimal Steiner trees in graphs. Int. J. Math. Educ. Sci. Technol. **14**(1), 15–23 (1983)
17. Rayward-Smith, V.J., Clare, A.: On finding Steiner vertices. Networks **16**(3), 283–294 (1986)
18. Segev, A.: The node-weighted steiner tree problem. Networks **17**, 1–17 (1987)
19. Siek, J.G., Lee, L., Lumsdaine, A.: The Boost Graph Library: User Guide and Reference Manual. Addison-Wesley Longman, Boston (2002)
20. Winter, P.: Steiner problem in networks: a survey. Networks **17**, 129–167 (1987)
21. Wong, R.: A dual ascent approach for Steiner tree problems on a directed graph. Math. Progr. **28**, 271–287 (1984)