Improved Variable Neighbourhood Search Heuristic for Quartet Clustering

Sergio Consoli[1,2(✉)], Jan Korst[2], Steffen Pauws[2,3], and Gijs Geleijnse[2,4]

[1] European Commission, Joint Research Centre, Directorate A-Strategy, Work Programme and Resources, Scientific Development Unit, Via E. Fermi 2749, 21027 Ispra, VA, Italy
sergio.consoli@ec.europa.eu
[2] Philips Research, High Tech Campus 34, 5656 AE Eindhoven, The Netherlands
[3] TiCC, Tilburg University, Warandelaan 2, 5037 AB Tilburg, The Netherlands
[4] Netherlands Comprehensive Cancer Organisation (IKNL), Zernikestraat 29, 5612 HZ Eindhoven, The Netherlands

Abstract. Given a set of n data objects and their pairwise dissimilarities, the goal of quartet clustering is to construct an optimal tree from the total number of possible combinations of quartet topologies on n, where optimality means that the sum of the dissimilarities of the embedded (or consistent) quartet topologies is minimal. This corresponds to an NP-hard combinatorial optimization problem, also referred to as minimum quartet tree cost (MQTC) problem. We provide details and formulation of this challenging problem, and propose a basic greedy heuristic that is characterized by a very high speed and some interesting implementation details. The solution approach, though simple, substantially improves the performance of a Reduced Variable Neighborhood Search for the MQTC problem. The latter is one of the most popular heuristic algorithms for tackling the MQTC problem.

Keywords: Combinatorial optimization · Quartet trees · Hierarchical clustering · Metaheuristics · Variable Neighbourhood Search · Graph theory

1 Introduction

Quartet clustering methods are popular in computational biology, where dendrograms (or phylogenies) are ubiquitous. These methods aim at reconstructing a rooted dendrogram from a set of pairwise distant objects (or taxa). Given a set of objects, define Q to be the set of all the possible quartets, and Q_t to be the set of consistent quartets being embedded in a dendrogram t. The problem of recombining the quartet topologies of Q to form an estimate of the correct tree diagram can be naturally formulated as an optimization problem. Steel [19]

A. Sifaleras et al. (Eds.): ICVNS 2018, LNCS 11328, pp. 1–12, 2019.
https://doi.org/10.1007/978-3-030-15843-9_1

formulated the *maximum quartet consistency (MQC) problem*, which looks for a dendrogram tree t maximizing the number of consistent quartets Q_t belonging to a subset $P \subseteq Q$ of quartet topologies. This problem has been shown to be NP-hard [19], and Jiang et al. [15] proved that the problem admits a polynomial time approximation scheme by using the technique of smooth integer polynomial programming and by exploiting the natural denseness of the set Q. However, this scheme only guarantees a dendrogram that may deviate from Q by εn^4 quartet topologies for any small constant $\varepsilon > 0$, where n is the number of taxa.

Due to these results, most quartet methods are heuristics which attempt to solve the MQC problem, or some variants of the MQC problem with weaker optimization requirements. Strimmer and von Haeseler [20] formulated the *quartet tree-puzzling problem*, which is a variant of the MQC problem where each quartet is provided with a probability value to be embedded, and for each set of four objects the quartet with the highest probability is selected (at random in case of ties) to form a "maximum-likelihood dendrogram". Felsenstein [11] presented a heuristic which solves the MQC by incrementally growing the tree diagram in random order by stepwise addition of objects in the local optimal way. This procedure is repeated iteratively for different object orders, adding agreement values on the branches of the tree. Both agglomerative approaches are quite fast, but suffer from the usual bottom-up problem: a wrong decision early on cannot be corrected later. Berry et al. [1] reported an interesting result. They presented two "quartet cleaning" algorithms for correcting bounded numbers of quartet errors (i.e. incorrect inferences of simple quartet topologies) for many popular quartet problems.

Cilibrasi and Vitányi [2] proposed for the first time the minimum quartet tree cost (MQTC) problem. Given a set N of $n \geq 4$ objects, the MQTC deals with a *full unrooted binary tree* with n leaves, a special topology dendrogram having all internal nodes connected exactly with three other nodes, the n objects assigned as leaf nodes, and without any distinction between parent and child nodes [12]. A full unrooted binary tree with $n \geq 4$ leaves has exactly $n - 2$ internal nodes, and consequently has a total of $2n - 2$ nodes. Full unrooted binary trees are of primary interest in clustering contexts because, of all tree diagrams with a fixed number of nodes, they have the richest internal structure (most differentiated paths between nodes). They are therefore very suitable for representing the structure of a set of objects [12]. A full unrooted binary tree with exactly $n = 4$ leaves is also referred to as *simple quartet topology*, or just as *quartet* [10,12]. Given a set N of $n \geq 4$ objects, the number of sets of four objects from the set N is given by:

$$\binom{n}{4} = \frac{n!}{4!(n-4)!} = \frac{n(n-1)(n-2)(n-3)}{24}.$$

Given four generic objects $\{a, b, c, d\} \in N$, there exist exactly three different quartets: $ab|cd$, $ac|bd$, $ad|bc$, where the vertical bar divides the two pairs of leaves, with each pair labelled by the corresponding objects and attached to the same internal node. Therefore the total number of possible simple quartet topologies of N is: $3 \cdot \binom{n}{4}$.

A full unrooted binary tree is said to be "consistent" with respect to a simple quartet topology, say $ab|cd$, *if and only if* the path from a to b does not cross the path from c to d. This quartet $ab|cd$ is also said to be "embedded" in the given full unrooted binary tree.

Considering the set N, the MQTC problem accepts as input a *distance matrix*, $\boldsymbol{D}$, which is a matrix containing the dissimilarities, taken pairwise, among the n objects[1]. To extract a hierarchy of clusters from the distance matrix, the MQTC problem determines a full unrooted binary tree with n leaves that visually represents the symmetric $n \times n$ distance matrix as well as possible according to a cost measure. Consider the set Q of all possible $3 \cdot \binom{n}{4}$ quartets, and let $C : Q \rightarrow \Re^+$ be a cost function assigning a real valued cost $C(ab|cd)$ to each quartet topology $ab|cd \in Q$. The cost assigned to each simple quartet topology is the sum of the dissimilarities (taken from $\boldsymbol{D}$) between each pair of neighbouring leaves [4]. For example, the cost of the quartet $ab|cd$ is $C(ab|cd) = D_{(a,\ b)} + D_{(c,\ d)}$, where $D_{(a,\ b)}$ and $D_{(c,\ d)}$ indicate, respectively, the dissimilarities among (a and b) and (c and d), obtained from the $\boldsymbol{D}$.

Consider now the set Γ of all full unrooted binary trees with $2n-2$ nodes (i.e. n leaves and $n-2$ internal nodes), obtained by placing the n objects to cluster as leaf nodes of the trees. For each $t \in \Gamma$, precisely one of the three possible simple quartet topologies for any set of four leaves is consistent [4]. Thus, there exist precisely $\binom{n}{4}$ consistent quartet topologies (one for each set of four objects) for each $t \in \Gamma$.

The cost associated with a full unrooted binary tree $t \in \Gamma$ is the sum of the costs of its $\binom{n}{4}$ consistent quartet topologies, that is: $C(t) = \sum_{\forall ab|cd \in Q_t} C(ab|cd)$, where Q_t is the set of such $\binom{n}{4}$ quartet topologies embedded in t.

In a hierarchical clustering context, we do not even have a priori knowledge that certain simple quartet topologies are objectively true and must be embedded. Thus, the MQTC problem assigns a cost value to each simple quartet topology, in order to express the relative importance of the simple quartet topologies to be embedded in the full unrooted binary tree having the n objects as leaves. The full unrooted binary tree with the minimum cost balances the importance of embedding different quartet topologies against others, leading to a binary tree that visually represents the symmetric distance matrix $n \times n$ as well as possible. The solution of this problem allows the hierarchical representation of a set of n objects within a full unrooted binary tree [12]. That is, the resulting binary tree will have the n objects assigned as leaves such that objects with short relative dissimilarities will be placed close to each other in the tree. This hierarchical clustering approach coming from the MQTC problem is also referred in the literature to as *quartet method* [4]. Such method is more sensitive and objective than other quartet clustering methods, which are usually too slow when they are exact or global, and too inaccurate or uncertain when they are statistical incremental, like the case of quartet tree-puzzling. In [4] the MQTC problem was shown to be NP-hard, and a Randomized Hill Climbing heuristic was also

[1] It is therefore a symmetric $n \times n$ matrix, with $n \geq 4$, containing non-negative reals, normalized between 0 and 1, as entries.

proposed to obtain approximate problem solutions. Other MQTC metaheuristics based on Greedy Randomized Adaptive Search Procedure, Simulated Annealing, and Variable Neighbourhood Search were proposed in [6]. These metaheuristics performed well for the problem, although the best performance was obtained by a Reduced Variable Neighbourhood Search (RVNS) implementation [6].

In this paper we propose some improved metaheuristics for the MQTC problem, to be used to get solutions of higher quality, in terms of reduced costs and computational running times. In particular we first propose a basic greedy heuristic which is characterized by a very high speed and some interesting implementation improvements, which can be used to enhance the MQTC metaheuristics to date in the literature. This greedy algorithm is characterized by its ease of implementation and simplicity, and it takes inspiration from the recently proposed "less is more approach" [9,18], which supports the adoption of non-sophisticated and effective metaheuristics instead of hard-to-reproduce and complex solution approaches. In particular we will show how the performance of the RVNS quartet heuristic is improved, with particular emphasis to computational running time, by adopting our proposed basic greedy heuristic to construct initial solutions for the algorithm.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 describes the details of the proposed heuristics, along with their main implementation concepts and pseudo-code formulations. Our computational experience is reported in Sect. 4, and finally the paper ends with conclusions in Sect. 5.

2 Related Work

The MQTC problem was originally proposed in [2]. There the main focus was on compression-based distances, but the authors visually presented the tree reconstruction results by full unrooted binary trees deriving by their MQTC problem formulation. Hence, they developed the quartet method for hierarchical clustering, a new approach aimed at general hierarchical clustering of data from different domains, not necessarily biological phylogenies. Several practical applications of the quartet method have been explored in the literature. In particular, Cilibrasi et al. [5] proposed a robust automatic music classification procedure consisting of two steps. The first step consisted of extracting the "Normalized Compression Distances" [16] among some considered pieces of music. The Normalized Compression Distance is a similarity metric based on string compression which mimics the ideal performance of Kolmogorov complexity [16]. The second step consisted of creating an efficient visualization of the extracted pairwise distances by means of the quartet method of hierarchical clustering. To substantiate the claims of universality and robustness of this automatic classification method, evidence of other successful applications in areas as diverse as genomics, virology, languages, literature, handwriting, astronomy and combinations of objects from completely different domains, were reported in [2]. In addition, Cilibrasi and Vitányi [3] reported an interesting application of this

theory, consisting of the automatic extraction of similarities among words and phrases from the WWW using Google page counts. Granados et al. [13] studied the impact of several kinds of information distortion on compression-based text clustering, showing their results as ternary trees by means of the quartet method of hierarchical clustering. In a recent application, a variant of the quartet method based on the Variable Neighborhood Search metaheuristic was used for biomedical literature extraction and clustering [7,8]. The proposed application was able to retrieve relevant references for systematic reviews and meta-analysis from the Medline/PubMed database, and for visualizing the retrieved bibliography through an intuitive graph layout.

In [4], the authors presented the minimum quartet tree cost problem in a more formal way. They showed the main concepts, components, advantages and disadvantages of the quartet method of hierarchical clustering, particularly underlining the similarities and differences with respect to other methods from biological phylogeny. Cilibrasi and Vitányi [4] also showed that the MQTC problem is NP-hard by reduction from the MQC problem, and provided a Randomized Hill Climbing heuristic to obtain approximate problem solutions. Several other efficient metaheuristics based on Greedy Randomized Adaptive Search Procedure, Simulated Annealing, and Variable Neighbourhood Search were proposed and compared for the MQTC problem in [6]. The best reported performance was obtained by an implementation of a Reduced Variable Neighbourhood Search metaheuristic, which we will use as a reference benchmark in our paper and try to overcome its performance.

3 Description of the Solution Algorithms

3.1 Greedy Constructive Heuristic

We first propose a new greedy heuristic for the MQTC problem, used to construct initial solutions of good quality requiring short computational running time [9, 18]. In the metaheuristics to date used for solving the MQTC problem [4,6], the initial solution was usually set either completely at random, or by selecting the corresponding flat structure, and then this solution was iteratively improved towards local optimality using the different heuristic guidelines of the specific metaheuristic implementation. The aim of the greedy constructive heuristic that we propose here consists of providing starting solutions having already a good quality, and obtained with an high speed too, which can bring to an improvement of the overall performance of the MQTC heuristic deployed afterwards.

We are given as input $n \geq 4$ different objects and the corresponding symmetric distance matrix $\boldsymbol{D}$ containing the $n \times n$ pairwise distances among those objects. The algorithm makes use also of another distance matrix $\boldsymbol{D'}$ among a set N' of $n' \geq 4$ objects, with $n' \leq n$, which will be used iteratively from our optimization routine to reduce the dimensionality n of the original set of objects in N. Initially, matrix $\boldsymbol{D'}$ is set equal to $\boldsymbol{D}$, i.e. the sets of objects N and N' are equivalent. At this stage, another graph t', which will be used during the algorithm iterations as a support solution, is initialized to null, i.e. $t' \leftarrow \emptyset$. Then the

core of our greedy heuristic begins by selecting objects from N' to be included in the support solution t'. At this purpose we greedily select from N' the objects that have the shortest minimum pairwise distance from $\boldsymbol{D'}$. Say the two objects a and b in N' have this shortest distance, that is $D'_{(a,\ b)} \leq D'_{(c,\ d)}, \forall (c,\ d) \in N'$. Note that in case of ties for the object pairs having the shortest distance in $\boldsymbol{D'}$, the routine simply selects an object pair at random within this set. Afterwards, these nodes a and b are connected to the support solution graph t'. The following three cases are possible:

- None of the two objects a and b are already connected in the partial solution t', and therefore they are joined together by means of a terminal node;
- One of the two objects is already included as a leaf node in t', and therefore the other object b is linked to the subgraph in t' containing a by means of a transition node (i.e. the dotted internal node in the figure). Please note that node b requires to be included in the partial solution t' by a link with a new transition node since, being it a leaf, if it would be included by a link with a terminal node instead, we would not be able to add any further nodes afterwards;
- Both objects a and b are already included in the partial solution t' but they belong to two different subgraphs, and therefore these two subgraphs containing respectively the two objects are connected together by means of a cross node.

Afterwards, we apply a routine, referred to as *distance matrix reduction*, to merge the added nodes a and b together to form another object, say x, by reducing in this way the dimension of N' of one unit, i.e. $n' = n' - 1$. The distance matrix D' is recomputed accordingly by removing the distances of the two objects a and b with all the other objects in N', and adding the distances of the new node x with the other objects, which are calculated as the averages distances, respectively of a and b, with the other nodes in N'. That is $D'(x,\ y) = \frac{D'(a,\ y)+D'(b,\ y)}{2}$, for all objects $y \in N'$, $y \neq a, b$. The rationale behind this procedure is to greedily merge together highly connected objects which may bring higher values of the quartet cost function of the subgraphs inferred in the partial solution t'.

This greedy procedure is repeated iteratively until a fully connected unrooted binary tree t' is obtained, i.e. $t' \in \Gamma$, which is equivalent also in getting a reduced distance matrix $\boldsymbol{D'}$ with a size $n' = 4$ (i.e. it would not be possible to reduce further the corresponding set N' since it only contain four objects). Then the support solution t' is assigned to the output full unrooted binary tree t, which is produced as final outcome of the algorithm.

3.2 Reduced Variable Neighbourhood Search

Variable Neighbourhood Search (VNS) is a popular metaheuristic for solving hard combinatorial optimization problems based on dynamically changing neighbourhood structures during the search process [14]. VNS does not follow a trajectory, but it searches for new solutions in increasingly distant neighbourhoods of

the current solution, jumping only if a better solution is found. Reduced Variable Neighbourhood Search (RVNS) is a variant of the classic VNS algorithm, that has been shown to be successful for many combinatorial problems where local optima with respect to one or several neighbourhoods are relatively close to each other [14]. RVNS is a typical example of a pure stochastic heuristic, akin to a classic Monte-Carlo method, but more systematic [17]. It is useful especially for very large problem instances for which the inner local search within the classic VNS approach is costly, as in the case with quartet clustering.

The Reduced Variable Neighbourhood Search for the MQTC problem starts by selecting an initial full unrooted binary tree $t \in \Gamma$ with $2n-2$ nodes, obtained by placing the $n \geq 4$ objects to cluster as leaves, with total cost $C(t)$. In the original RVNS implementation in [6], the initial full unrooted binary tree t was selected at random.

Then, the *shaking phase*, which represents the core idea of RVNS, is applied to t. The shaking phase aims to change the neighbourhood structure, $N_k(\cdot)$, when the algorithm is trapped at a local optimum. The new incumbent solution, say t', is generated at random in order to avoid cycling, which might occur if a deterministic rule is used. The simplest and most common choice for the neighbourhood structure consists of setting neighbourhoods with increasing cardinality: $|N_1(\cdot)| < |N_2(\cdot)| < ... < |N_{k_{max}}(\cdot)|$, where k_{max} represents the maximum size of the shaking phase. Let k be the current size of the shaking phase. The algorithm starts by selecting the first neighbourhood ($k \leftarrow 1$) and, at each iteration, it increases the parameter k if a better solution is not obtained ($k \leftarrow k+1$), until the largest neighbourhood is reached ($k \leftarrow k_{max}$). The process of changing neighbourhoods when no improvement occurs diversifies the search. In particular, the choice of neighbourhoods of increasing cardinality yields a progressive diversification of the search process.

For the MQTC problem, a shaking phase of size k consists of the random selection of another full unrooted binary tree t' within the neighbourhood $N_k(t)$ of the current solution t. To obtain t' from $N_k(t)$, the algorithm performs k consecutive *base moves*, where a base move is a single basic modification that each internal node of t can perform with its neighbouring internal nodes. The possible base moves that can be performed depend on the types of internal node pairs [6]. In the case of:

- two transition nodes: either the attached leaves are exchanged, or they are transformed into one cross node and one terminal node connected to the corresponding leaves;
- one terminal node and one transition node: the leaf of the transition node is exchanged with one of the two leaves of the terminal node;
- one terminal node and one cross node: they are transformed into two transition nodes with the two leaves of the terminal node attached;
- one transition node and one cross node: the transition node is moved in one of the other two branches of the cross node;
- two cross nodes: one branch of one cross node is swapped with a branch of the other cross node.

Note that each base move corresponds just to a limited local modification of the structure of the incumbent solution t, which results in most of the coefficients of the corresponding Complete Pseudo-Adjacency matrix $\boldsymbol{C}$ to remain unchanged. In this was there will be no need to recalculate all the coefficients of $\boldsymbol{C}$, but only recomputing a small subset of it, speeding up consistently this step.

At the beginning of RVNS, the first neighbourhood ($k \leftarrow 1$) is selected and, at each iteration, the parameter k is increased ($k \leftarrow k+1$) whenever the solution obtained is not an improvement of the current best solution (i.e. $C(t') > C(t)$). When $k > 1$, the first base move is performed to a randomly selected internal node and one of its neighbouring internal nodes with respect to the considered distance of rank one. Then, to perform the successive base move, the algorithm selects one of the two internal nodes considered, and another neighbouring internal node that must be different from the two internal nodes already considered, and so on. The procedure is repeated until k consecutive base moves are performed.

If an improved binary tree t' is produced by the shaking phase ($C(t') < C(t)$), this becomes the best solution to date ($t \leftarrow t'$) and the algorithm restarts from the first neighbourhood ($k \leftarrow 1$) of t. The process of increasing progressively parameter k whenever no improvements are obtained, occurs until the maximum size of the shaking phase, k_{max}, is reached. When this happens, k is re-initialized to the first neighbourhood ($k \leftarrow 1$). The correct setting of k_{max} is an important user task. For the MQTC problem, a simple reactive schema for the efficient tuning of k_{max} has been implemented [6]. At the starting point, k_{max} is set to a small value ($k_{max} = 2$) and is increased ($k_{max} = k_{max}+1$) every i_{update} iterations between two consecutive improvements. For the value of this parameter we use the setting of [6], where $i_{\text{update}} = (1.25 \cdot 10^5)/n^2 + 50$. Throughout the execution of the algorithm, the best solution to date is stored as the binary tree t, which will be produced as output of the algorithm when the user termination condition (e.g. a maximum allowed CPU time) is reached.

4 Computational Results

In order to evaluate the algorithms, we performed experiments to compare them in terms of quality of produced solutions and computational running time. For evaluating solution quality, we used both the cost function $C(\cdot)$, already defined previously, and of another metric, referred to as *normalized tree benefit score*, $S(\cdot) \in [0, 1]$ [2,6], which is a more intuitive performance measure of the goodness of quartet clustering. Given the set N of $n \geq 4$ documents to cluster, let m be the *best (minimal) cost*, calculated as the sum of the $\binom{n}{4}$ minimum costs of each set of four objects in N, and let M be the *worst (maximal) cost*, calculated as the sum of the $\binom{n}{4}$ maximum costs of each set of four objects in N. The normalized tree benefit score $S(t)$ of a full unrooted binary tree $t \in \mathit{\Gamma}$ is obtained by rescaling and normalizing in $[0, 1]$ the cost function $C(t)$, i.e. $S(t) = \frac{M - C(t)}{M - m} \in [0, 1]$. While a lower cost function $C(t)$ results in a better solution t, conversely a higher normalized tree benefit score means a better clustering quality.

Our experimental algorithms comparison was made upon classic MQTC problem datasets, already used in previous studies in the literature (see e.g. [2,4,6]). They are briefly described in the following, but for more details the reader is referred to [2,4,6].

- Data constructed artificially to have none inconsistency, that is data for which the exact solutions are known in advance and have been built to have normalized tree benefit score equal to one. The construction mechanism is described in detail in [2,6]. These data aim at testing whether the quartet-based tree reconstruction is reliable and accurate on clean consistent data with known solutions. They consist of ten different problem instances ranging from a number of objects $n = 10$ to 100. They are referred to as: *artificial*.
- Example of natural data concerning a study in genomics with DNA sequences of different placental mammalian species. The distance matrices from the genomic data were computed by using an automated software method described by Cilibrasi and Vitányi [2,4], who downloaded the whole mitochondrial genomes of the placental mammalian species from the GenBank Database on the World Wide Web. They consists of three sets of data with $n = 10$, $n = 24$, and $n = 34$, and are referred to as: *nature*.

Table 1 show the results of our experimental comparison of the algorithms on the considered datasets. The heuristics are identified with the following abbreviations: *Greedy*, for the greedy constructive heuristic described in Sect. 3.1; $RVNS_{rand}$, for the original implementation of the Reduced Variable Neighbourhood Search (Sect. 3.2) with initial solution selected at random; $RVNS_{greedy}$, for the new Reduced Variable Neighbourhood Search implementation where the initial solution is selected by using the greedy constructive heuristic, *Greedy*. All the algorithms were implemented in C++ under the Microsoft Visual Studio 2015 framework, and were deployed on an Intel Quad-Core i5 64-bit microprocessor at 2.30 GHz with 16 GB RAM.

As stopping condition for the RVNS-based metaheuristics it was considered a maximum allowed CPU time (*max-CPU-time*). In particular, as also used in [2,6], we set *max-CPU-time* to one hour. Selection of the maximum allowed CPU time as the stopping criterion was made in order to have a direct comparison among the RVNS metaheuristics with respect to the quality of their solutions. Instead, for the *Greedy* algorithm it was not necessary to set any stopping criterion since, being a constructive heuristic, it automatically ends when a feasible solution, i.e. a fully connected unrooted binary tree, is obtained.

Looking at Table 1, the first column shows the number of objects, n, characterizing the different datasets (*artificial*, *nature*, *geographical*) while the remaining columns give the computational results in terms of clustering quality (i.e. cost function values $C(\cdot)$, *cost*, and normalized tree benefit scores $S(\cdot)$, *score*), and computational running time in seconds (*time*) for the different algorithms. The performance of an heuristic can be considered better than another if it obtains a lower cost function value, or more intuitively a larger normalized tree benefit score. In case of ties, an algorithm is consider better than another if it was faster.

Table 1. Computational results of the compared algorithms (*Greedy*, $RVNS_{rand}$, and $RVNS_{greedy}$) in terms of cost function values (*cost*, normalized tree benefit scores (*score*, and computational running times in seconds (*time*) for the considered datasets.

	Greedy			$RVNS_{rand}$			$RVNS_{greedy}$		
size n	*cost*	*score*	*time*	*cost*	*score*	*time*	*cost*	*score*	*time*
artificial									
10	210.8000	0.89500	0.004	202.4000	1.00000	0.040	202.4000	1.00000	0.005
20	3301.2500	0.95270	0.003	3231.8500	1.00000	0.421	3231.8500	1.00000	0.006
30	15013.8656	0.92419	0.002	14559.4658	1.00000	0.861	14559.4658	1.00000	0.023
40	45245.7000	0.92400	0.006	43449.7500	1.00000	8.413	43449.7500	1.00000	0.554
50	104709.0800	0.85641	0.008	97207.8400	1.00000	10.606	97207.8400	1.00000	0.055
60	196608.3789	0.89166	0.012	186787.0474	1.00000	38.724	186787.0474	1.00000	0.176
70	360831.5064	0.88295	0.022	338182.8198	1.00000	38.858	338182.8198	1.00000	0.243
80	561067.4125	0.79739	0.028	509526.6875	1.00000	66.880	509526.6875	1.00000	0.193
90	806291.9392	0.89048	0.037	769344.2770	1.00000	101.512	769344.2770	1.00000	0.287
100	1232141.4400	0.92249	0.059	1178538.2000	1.00000	115.013	1178538.2000	1.00000	10.292
nature									
10	349.0720	0.99979	0.002	349.0720	0.99979	0.006	349.0720	0.99979	0.039
24	18649.3360	0.98524	0.002	18637.3390	0.99588	2.083	18637.3390	0.99588	0.232
34	82934.2444	0.98323	0.005	82922.0360	0.98792	10.610	82922.0360	0.98792	0.542
geographical									
13	476.4124	0.74265	0.002	439.7529	0.96843	0.270	439.7529	0.96843	0.016
22	4644.7111	0.82426	0.004	4377.1741	0.93507	3.140	4377.1741	0.93507	0.029
24	6839.0909	0.79911	0.004	6422.0182	0.92459	3.290	6422.0182	0.92459	0.044
25	8876.1814	0.79267	0.004	7827.1264	0.98760	2.840	7827.1264	0.98760	0.046
35	29209.7541	0.81774	0.004	26332.7485	0.98367	10.750	26332.7485	0.98367	0.205
37	28298.8724	0.79559	0.006	26846.2316	0.91973	32.940	26846.2316	0.91973	0.318

From the results showed in the table, we can immediately denote that *Greedy* was much faster of several orders of magnitude than the original RVNS implementation, $RVNS_{rand}$, although in most of the cases it obtained solutions with worst quality. This is an understandable result since $RVNS_{rand}$ is an explorative metaheuristic that runs for a longer time, *max-CPU-time*, while *Greedy* instead stops immediately when a feasible solution is reached. But when *Greedy* was then embedded inside the RVNS metaheuristic in order to produce initial good-quality solutions in $RVNS_{greedy}$, a very powerful metaheuristic was obtained. Indeed, as it can be seen in the table, $RVNS_{greedy}$ retained the high-speed feature from *Greedy*, but also the characteristics of good-quality solutions that is proper of the RVNS approach for the given problem. Indeed, looking at the performance of both the RVNS algorithms, the obtained solutions were comparable with respect to clustering quality, but $RVNS_{greedy}$ was much faster. Note that for the *artificial* datasets without inconsistencies, both RVNS implementations were able to reach optimality, $RVNS_{greedy}$ being much faster, while this was not achieved by *Greedy*.

Summarizing, the novel RVNS implementation with the greedy constructive heuristic used for selecting the initial starting solutions resulted to be the best performing method in our computational experiments in terms of both quartet clustering quality and, especially, computational running time.

5 Conclusions

In this paper we proposed some improved heuristics for quartet clustering, a novel hierarchical clustering approach based on the minimum quartet tree cost (MQTC) problem, which is NP-hard and whose goal is to derive an optimal tree from the total number of possible combinations of quartet topologies on some input objects n, where optimality means that the sum of the dissimilarities of the embedded (or consistent) quartet topologies is minimal.

In particular we provided the details of a new basic greedy heuristic that is characterized by a very high speed. Although the performance of this simple method in terms of quartet clustering quality, evaluated by means of a defined cost function and of a normalized tree benefit score, was not as good as that of the best solution method reported in the literature, i.e. a Reduced Variable Neighbourhood Search (RVNS) metaheuristic, this greedy method was used to considerably improve the performance of the RVNS by using it to construct initial good-quality solutions instead of randomly selected solutions.

This produces a very efficient solution approach to the problem, as demonstrated by our experiments on the comparison of the considered algorithms on a set of well-known MQTC datasets and by the reported computational results, which represents an advancement of the state-of-the-art on the solution methods used for quartet clustering.

Acknowledgements. The author Dr. Sergio Consoli wants to dedicate this work with deepest respect to the memory of Professor Kenneth Darby-Dowman, a great scientist, an excellent manager, the best supervisor, a wonderful person, a real friend.

References

1. Berry, V., Jiang, T., Kearney, P., Li, M., Wareham, T.: Quartet cleaning: improved algorithms and simulations. In: Nešetřil, J. (ed.) ESA 1999. LNCS, vol. 1643, pp. 313–324. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48481-7_28
2. Cilibrasi, R., Vitányi, P.M.B.: Clustering by compression. IEEE Trans. Inf. Theory **51**(4), 1523–1545 (2005)
3. Cilibrasi, R., Vitányi, P.M.B.: The google similarity distance. IEEE Trans. Knowl. Data Eng. **19**(3), 370–383 (2007)
4. Cilibrasi, R., Vitányi, P.M.B.: A fast quartet tree heuristic for hierarchical clustering. Pattern Recogn. **44**(3), 662–677 (2011)
5. Cilibrasi, R., Vitányi, P.M.B., de Wolf, R.: Algorithmic clustering of music based on string compression. Comput. Music J. **28**(4), 49–67 (2004)
6. Consoli, S., Darby-Dowman, K., Geleijnse, G., Korst, J., Pauws, S.: Heuristic approaches for the quartet method of hierarchical clustering. IEEE Trans. Knowl. Data Eng. **22**(10), 1428–1443 (2010)
7. Consoli, S., Stilianakis, N.I.: A VNS-based quartet algorithm for biomedical literature clustering. Electron. Notes Discrete Math. **47**, 13–20 (2015)
8. Consoli, S., Stilianakis, N.I.: A quartet method based on variable neighborhood search for biomedical literature extraction and clustering. Int. Trans. Oper. Res. **24**(3), 537–558 (2017)

9. Costa, L.R., Aloise, D., Mladenović, N.: Less is more: basic variable neighborhood search heuristic for balanced minimum sum-of-squares clustering. Inf. Sci. **415–416**, 247–253 (2017)
10. Diestel, R.: Graph Theory. Springer, New York (2000)
11. Felsenstein, J.: Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. **17**(6), 368–376 (1981)
12. Furnas, G.W.: The generation of random, binary unordered trees. J. Classif. **1**(1), 187–233 (1984)
13. Granados, A., Cebrian, M., Camacho, D., Rodriguez, F.B.: Reducing the loss of information through annealing text distortion. IEEE Trans. Knowl. Data Eng. **23**(7), 1090–1102 (2011)
14. Hansen, P., Mladenović, N.: Variable neighbourhood search. Comput. Oper. Res. **24**, 1097–1100 (1997)
15. Jiang, T., Kearney, P., Li, M.: A polynomial time approximation scheme for inferring evolutionary trees from quartet topologies and its application. SIAM J. Comput. **30**(6), 1942–1961 (2000)
16. Li, M., Vitányi, P.M.B.: An Introduction to Kolmogorov Complexity and Its Applications, 2nd edn. Springer, New York (1997)
17. Mladenović, N., Petrović, J., Kovačević-Vujčić, V., Čangalović, M.: Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. Eur. J. Oper. Res. **151**(2), 389–399 (2003)
18. Mladenović, N., Todosijević, R., Urošević, D.: Less is more: basic variable neighborhood search for minimum differential dispersion problem. Inf. Sci. **326**, 160–171 (2016)
19. Steel, M.A.: The complexity of reconstructiong trees from qualitative characters and subtrees. J. Classif. **9**, 91–116 (1992)
20. Strimmer, K., von Haeseler, A.: Quartet puzzling: a quartet maximum-likelihood method for reconstructing tree topologies. Mol. Biol. Evol. **13**(7), 964–969 (1996)