An Improved Exact Algorithm for a Territory Design Problem with p-Center-Based Dispersion Minimization [1]

M. Gabriela Sandoval
Department of Computers, Electronics and Mechatronics
Universidad de las Américas Puebla (UDLAP)
San Andrés Cholula, Puebla 72820, Mexico
maria.sandovalel@udlap.mx

Juan A. Díaz
Department of Actuary, Physics and Mathematics
Universidad de las Américas Puebla (UDLAP)
San Andrés Cholula, Puebla 72820, Mexico
juana.diaz@udlap.mx

Roger Z. Ríos-Mercado
Graduate Program in Systems Engineering
Universidad Autónoma de Nuevo León (UANL)
San Nicolás de los Garza, Nuevo León 66455, Mexico
roger.rios@uanl.edu.mx

April 2019

[1]Technical report PISIS-2019-01, Graduate Program in Systems Engineering, UANL, San Nicolás de los Garza, NL, Mexico, April 2019

Abstract

Territory design deals with the discrete assignment of geographical units into territories with restrictions defined by planning criteria. We propose an exact method of solution based on an integer programming model with the objective of minimizing a p-center dispersion measure. The solution approach is an iterative algorithm that uses different subproblems to validate if, for given values of the objective function of the original problem, it is possible to find feasible solutions with at most p territories. This change allows testing various candidate distance values as lower bounds for the optimal solution of the original problem. The aim is to improve these lower bounds at each iteration as we add the necessary constraints to reach a feasible solution. The proposed algorithm performs significantly faster than existing methods with small and medium-sized instances.

Keywords: Territory design; p-center problem; integer programming; model reformulation.

1 Introduction

Territory design problems (TDPs) deal with the division of a geographical area into territories in line with planning requirements. They arise in several social planning contexts and their aim is to simplify management over the area by focusing on each territory individually. These problems have been studied in the area of Operations Research since the 1960's and research on this type of problems may also be found under the name of districting problems or territory alignment. Most of the research done in the early years falls under two main lines of study: political districting and sales and service territory design. The former focuses on the fair division of an area for either democratic representation or governance. [22] have reviewed in detail the research done on political districting. The latter deals with the design of regions of responsibility for either salesmen or managers over the areas where a business operates. The most prominent work on such application is attributed to Zoltners and Sinha [32] who have worked on this type of problems for over three decades. The reader is referred to the surveys by Kalcsics and Ríos-Mercado [16] and Duque et al. [6] for a good overview of different approaches developed for TDPs.

The commercial territory design problem with p-center based minimization criterion, which is the focus of this paper, was introduced by [24]. The problem consists of minimizing territory dispersion based on the p-center measure while balancing multiple activity measures and ensuring territory connectivity. This problem was motivated by a real-case application of a beverage distribution company. The authors presented a GRASP heuristic for this problem focusing on large scale instances (with up to 500 nodes). To the best of our knowledge, the only exact method developed for this problem is due to [27]. In their work, the authors present a solution framework based on branch and bound and a cut generation strategy. Their computational results reported optimal solutions for instances with up to 150 nodes.

In this paper, we present an exact optimization method for this problem that uses auxiliary sub-problems to obtain lower bounds for the distance value that minimizes the dispersion objective. This process is an iterative algorithm inspired by the procedure developed by Özsoy and Pınar [19] for the capacitated p-center problem. The algorithm exploits the similarities between both problems and the fact that there is a finite list of candidate values for the dispersion objective which makes it suitable for inspection. The proposed algorithm also includes a strategy to reduce the computational burden of the connectivity constraints as the one presented by Salazar-Aguilar et al. [27]. The empirical evidence indicates the proposed algorithm significantly outperforms the existing approach in terms of average running time. The success of the algorithm comes from the use of the auxiliary sub-problems that have a significantly reduced number of decision variables.

The rest of the paper is organized as follows. Section 2 contains a brief literature survey on TDPs. Section 3 includes the description of the problem and the mathematical model. Section 4 includes a detailed explanation of the proposed method. Section 5 describes the evaluation procedure and the results of the tests. Finally, Section 6 includes the concluding remarks.

2 Literature Survey

In recent years, TDPs have gained increasing interest as new applications have arisen and sophisticated methods of solution have been developed. Some of the novel applications include the design of service areas for: schools [9] , police stations [5, 3, 4], earthquake shelters [14], population settlement lots [11], access to liver transplants [10], and recollection of waste electric and electronic equipment [8].

Research on TDPs is commonly application oriented; however, as noted by Kalcsics et al. [15], most applications share the objective of creating balanced territories with a compact and connected shape. In the literature one can identify different motivations behind these requirements for TDPs as well as different strategies to model and ensure them.

Balanced territories are of similar size according to one or more activity measures and it enables the generalization of management strategies over territories. In political districting, for example, it is desired to have territories with about the same amount of voters and an even distribution of minorities [22]. In the case of sales and service territory design balance is often related to an even distribution of workload which can be measured with number of clients or quantity of demand [12]. In relation to the design of service areas for the novel applications, the aim is to have territories with balanced demand for the service and the measure for demand is different according to the application.

The shape for a territory can potentially make its management easier by simplifying travel routes within each territory and preventing unwanted influence between territories (or gerrymandering). The shape of a territory in most TDPs is characterized by constraints of connectivity and compactness. In a connected territory it is possible to travel between any two points in the territory without leaving the territory. Ensuring connectivity is sometimes a challenging task, [30] gives some insight on how to incorporate connectivity constraints to linear programming models.

The definition of a compact territory is more ambiguous, but it can be described as having its components close together and with a shape that is as round as possible without any holes. As noted by Shirabe [29] there is no consensus in the literature on the most adequate way to measure compactness. It is common to use dispersion measures such as moment of inertia [13], p-median [28, 25], p-center measures [7, 24] or diameter-based measures [23, 30]. Alternative strategies to ensure compactness include the use of convex hulls [2] and Voronoi regions [21] to generate inherently compact shapes for territories with measures that are not scale dependent.

The existing solution methods for TDPs can be divided into two main categories: those that obtain an exact solution for a mathematical programming model and those that use heuristics. The former includes formulations derived from location problems as well as set partitioning problems.

Some recent publications of these methods include Shirabe [30] and Salazar-Aguilar et al. [28] who have introduced models adapted from well studied location problems (such as the p-center or the p-median problem) to the requirements of TDPs. Lari et al. [17] and Ahuja et al. [1] propose several models that focus on viewing TDPs as set partitioning problems. Conversely, methods that only use heuristics include metaheuristic approaches and geometric algorithms. [11] and [14] have implemented a tabu search and genetic algorithm respectively for TDPs. An example of geometric algorithms is the work by [2] who proposes several geometric approaches to tackle TDPs.

There are also many successful heuristic implementations developed in the past for many kinds of territory design problems. Some of these include, [11] who applied a Tabu Serach algorithm for a TDP in commercial territory design, [14], who applied a sorting genetic algorithm for the planning of service areas, and [2], who presents several geometric algorithms in his doctoral thesis for the general model of TDPs, to name a few. For a more extensive discussion in solution algorithms for diverse TDPs the reader is referred to [16].

In our work, we focus on a commercial TDP which ensures compactness by minimizing a p-center dispersion measure while considering constraints of connectivity and multiple balance for an even distribution the number of clients and product demand. To the best of our knowledge, the only known methods that have tackled the same problem are by [24] who developed a metaheuristic approach and [27] who developed the first known exact method. Our contribution is the development of an exact algorithm that performs better than the existing approach that can be used for tackling larger instances.

3 Problem Description

For the discrete formulation of the TDP we consider the geographical area of study as an undirected graph (as the one shown in Figure 1) where nodes represent basic units (BUs) and edges connect BUs that are neighbors. Weights associated to nodes indicate the size of BUs according to each activity measure (such as product demand, workload, and number of customers), while weights associated to edges indicate distance values between neighboring BUs. In this sense, the TDP is viewed as the problem of finding a partition of the set of nodes into a given number of subsets (territories) that comply with certain planning criteria.

The specific criteria considered in this work are (i) balanced territories with respect to each of several activity measures, (ii) territory connectivity (or contiguity), and (iii) territory compactness. Territory balance means having territories of about the same size in relation to each activity measure. In turn, the size of a territory with respect to a given activity is simply the sum of the activity values of the individual basic units contained in that territory. The connectivity requirement is achieved by imposing that each territory induces a connected subgraph from the original graph. An example of an adequate graph partition in respect to connectivity is pictured in Figure

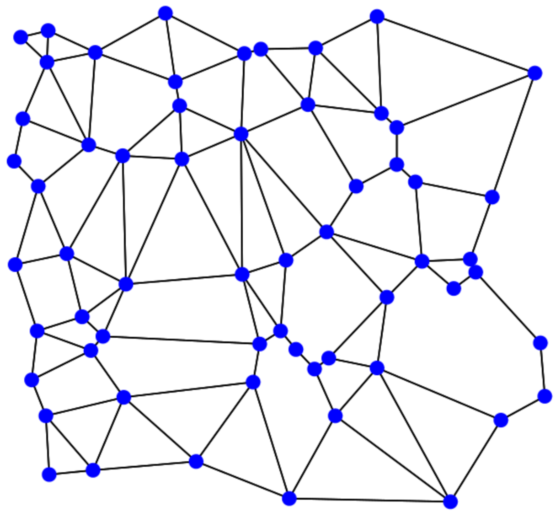


Figure 1: Example graph that represents a geographical area.

2. In the figure, nodes that belong to the same territory are associated with a specific color. The compactness issue is addressed by minimizing an adequate dispersion function. In this case we consider the p-center dispersion which requires the definition of one of the nodes in each territory as a center. The p-center measure is equal to the greatest distance between a territory center and the farthest BU assigned to it. This problem is refered to as the CTDP and was introduced by Ríos-Mercado and Fernández [24].

To formulate this problem, let G(V,E) be an undirected graph where V is the set of BUs that represents the geographical area of study and E the set of edges that connect neighboring BUs. Also, let A be the set of activity measures that are considered for the balance constraints of the practical problem. In this case we consider two activity measures: number of clients and product demand.

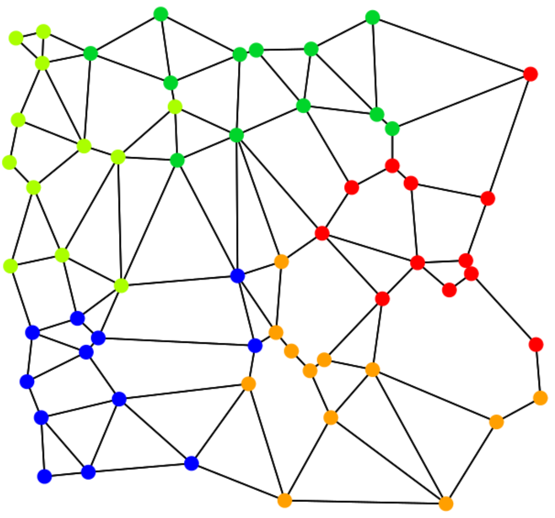


Figure 2: Example graph of an instance with a feasible partition into territories.

In addition, the following parameters are known:

- d_{ij}: Euclidean distance between a pair of BUs $i, j \in V$.
- $N^i \subseteq V$: set of adjacent neighbors of BU $i \in V$.
- w_i^a: the value of activity measure $a \in A$ that corresponds to BU $i \in V$.

Let x_{ij} be the binary decision variable that indicates if BU j belongs to the territory centered at BU i ($x_{ij} = 1$) or not ($x_{ij} = 0$). Note that the case $i = j$, ($x_{ii} = 1$) indicates that BU i is a territory center. The model is given by

(CTDP)

$$\text{minimize} \quad z = \max_{i,j \in V}\{d_{ij} x_{ij}\} \tag{1}$$

$$\text{subject to} \quad \sum_{i \in V} x_{ii} = p \tag{2}$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V \tag{3}$$

$$\sum_{j \in V} w_i^a x_{ij} \geq \frac{(1 - \tau^a)}{p} \Big(\sum_{j \in V} w_j^a \Big) x_{ii} \qquad i \in V, \quad a \in A \tag{4}$$

$$\sum_{j \in V} w_i^a x_{ij} \leq \frac{(1 + \tau^a)}{p} \Big(\sum_{j \in V} w_j^a \Big) x_{ii} \qquad i \in V, \quad a \in A \tag{5}$$

$$\sum_{j \in R(S)} x_{ij} - \sum_{j \in S} x_{ij} \geq 1 - |S| \qquad i \in V, \quad S \subset [V \setminus (N^i \cup \{i\})], \tag{6}$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in V \tag{7}$$

The objective function of this minimization problem is defined by (1). It corresponds to the p-center dispersion measure that, in the scope of the TDP problem, defines the maximum distance between a basic unit and the territory center it is assigned to. By minimizing this distance, compactness is maximized within each territory since this will result in BUs that are closer to its center. Constraint (2) limits the number of territories to be exactly p while constraints (3) ensure the unique assignment of each BU to a territory center. The balance constraints are described by (4) and (5). These constraints state that the size of a territory per activity measure a lies within a given deviation τ^a from the average size of a territory. This deviation tolerance parameter is user defined with typical values between 0.0 and 0.2, depending on the particular context. Connectivity is ensured with constraints (6). These constraints are similar to the sub-tour elimination constraints for the traveling salesman problem. For every territory center i two sets S and $R(S)$ are defined as S being any subset of the BUs assigned to i (not containing i nor any of its neighbors), and $R(S)$ being the subset of all the neighbors of S.

$$R(S) = \bigcup_{v \in S} (N^v \setminus S)$$

These constraints state that every territory center i must be connected to any subset of nodes (non-neighbors of i) assigned to that territory. Finally, (7) state the nature of the binary values.

As stated by [24], this problem is NP-hard given that it can be viewed as an extension of the capacitated vertex p-center problem which is NP-hard as well. Also, it is important to notice that there is an exponential number of connectivity constraints that makes the CTDP intractable for medium-sized instances.

4 Proposed Exact Algorithm

In this paper we propose an exact method of solution for the CTDP that is an iterative algorithm inspired by the methodology introduced by Özsoy and Pınar [19] for the capacitated vertex p-center problem (CVPC). They use an auxiliary subproblem to find an initial lower bound for the objective function and progressively add constraints to the subproblem to increase the lower bound. The algorithm ends when a subproblem gives a solution that is also feasible for the CVPC and thus the optimal solution is obtained. A similar methodology has been successfully implemented for other partition problems such as the capacitated controller placement problem [31] and the clusterhead placement problem [20]. Our method is tailored to deal with the constraints that differentiate the CTDP from the CVPC, namely, the ones that ensure territory balance and connectivity.

It is important to notice that in the CTDP the optimal value of the objective function will correspond to a distance value $z^* = d^*_{ij}$ between a certain pair of BUs. This means that the range of the objective function is defined by the set of different distance values between every pair of BUs in an instance of the problem. Such set is finite since there is also a finite number of BUs in every instance. Therefore, it is viable to test candidate distances in search of the optimal value of the CTDP. Let $D = \{d_1, d_2, ..., d_K\}$ be the different values from the distance matrix between pairs of BUs, such that $d_1 < d_2 < ... < d_K$. This list defines the search space of our algorithm that will be reduced at every iteration.

Testing candidate distance values for the objective of the CTDP implies of course the reformulation of the problem. Let us define the following integer programming (IP) subproblem SP(δ), where δ is a candidate distance value to be tested. The subproblem minimizes the number of territories for the fixed value (δ) of the p-center dispersion measure with constraints that are similar to the CTDP as described below. SP(δ) considers only the decision variables x_{ij} that correspond to BUs that have a distance value between them that is at most δ ($x_{ij}|d_{ij} \leq \delta$). By doing so, the decision variables that correspond to BUs having a distance value greater than δ are eliminated from the model. Thus, the size of this problem (or its feasible space) is reduced in comparison to the CTDP as a function of δ.

In order to test how different is a candidate δ from the objective of the CTDP, the optimal value (p^*) obtained by solving SP(δ) is compared with the desired number of territories (p) of the CTDP. If $p^* > p$ the candidate δ is lower than the optimal distance value of CTDP. This is true because by limiting the maximum distance between any BU and its territory center with δ, more than p territories are necessary to assign all BUs in line with the constraints of the subproblem. Figures 3a and 3b illustrate the number of territories formed when using a low and large value of delta, respectively. As can be seen, using a smaller value of delta results in the formation of more territories.

In pursuit of getting closer to the optimal value of the CTDP, if $p^* > p$, the next candidate value to be tested must be greater than δ with the aim of reducing the number of territories needed and thus getting closer to p. If for some value of δ the subproblem is infeasible the next candidate value to be tested must also be greater than δ in order to fit more BUs per territory and comply with the lower bounds for the balance constraints. Conversely, if $p^* < p$, the candidate δ is greater than the optimal distance value of CTDP because less than p territories were needed to assign the BUs in the graph. These rules guide the search of our algorithm over the list of distances between BUs for the optimal value of the CTDP.

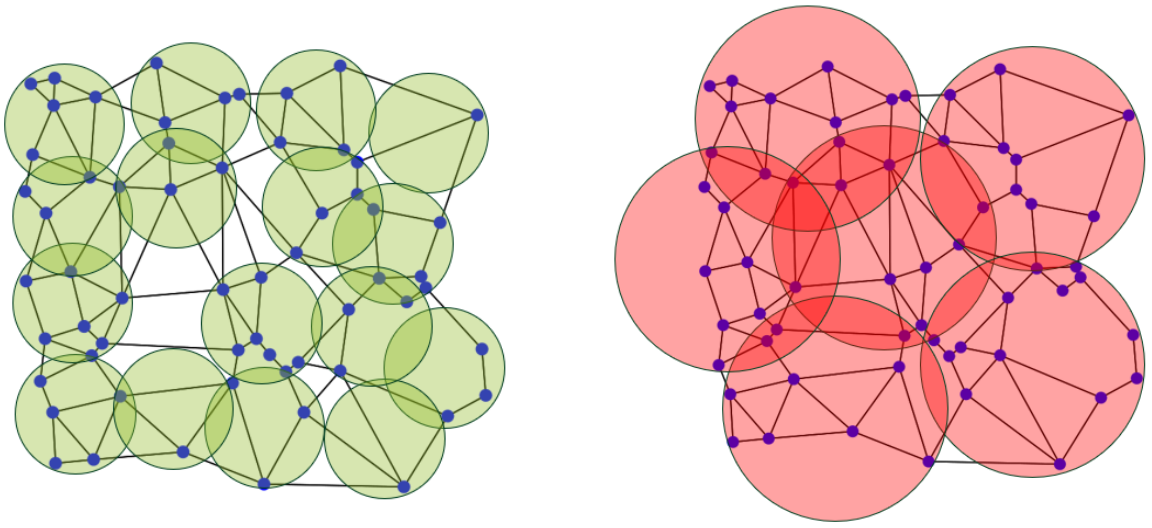

(a) (b)

Figure 3: Example of solutions of SP(δ). (a) small value of δ; (b) large value of δ. (Circles are used to show the maximum p-center dispersion value that corresponds to the value of δ in the subproblem)

(SP(δ))

$$\underset{x}{\text{minimize}} \quad z = \sum_{i \in V} x_{ii} \tag{8}$$

$$\text{subject to} \quad \sum_{i \in V_j(\delta)} x_{ij} = 1 \qquad j \in V \tag{9}$$

$$\sum_{j \in V_j(\delta)} w_i^a x_{ij} \geq \frac{(1-\tau^a)}{p} \Big(\sum_{j \in V} w_j^a \Big) x_{ii} \qquad i \in V, \quad a \in A \tag{10}$$

$$\sum_{j \in V_j(\delta)} w_i^a x_{ij} \leq \frac{(1+\tau^a)}{p} \Big(\sum_{j \in V} w_j^a \Big) x_{ii} \qquad i \in V, \quad a \in A \tag{11}$$

$$x_{ij} \leq x_{ii} \qquad i, j \in W(\delta) \tag{12}$$

$$x_{ij} \in \{0, 1\} \qquad i, j \in V \tag{13}$$

Subproblem SP(δ) (defined by (8) to (13)) is related to the CTDP model as follows. First, the connectivity constraints (6) are relaxed. Then, the dispersion minimization function (1) is replaced by (8) to minimize the number of territories. In addition, constraints (9) to (11) differ from constraints constraints (3) to (5) of the CTDP in considering only the decision variables x_{ij} that correspond to BUs that have a distance value between them that is at most δ ($x_{ij} | d_{ij} \leq \delta$). To do so we have defined set V$_j(\delta)$ as the set of BUs that have a distance to BU j lower than δ

and set W(δ) as the set of pairs of BUs that have a distance between them lower than δ.

$$V_j(\delta) = \{i \in V| \quad d_{ij} < \delta\}$$

$$W(\delta) = \{(i,j) \in V \times V| \quad d_{ij} < \delta\}$$

This model also includes constraints (12) to prevent a BU to be assigned to a territory center that is not open. Although these constraints are redundant in the problem formulation, they help improve the lower bound of the linear programming (LP) relaxation of the problem that is used in the algorithm. It is important to emphasize that if for some value of δ the subproblem resulted infeasible, then such value is smaller than the optimal value of the CTDP. This is due to the fact that an infeasibility in SP(δ) is attributed to not having enough BUs in a territory to reach a lower bound of the balance constraints in (10). A larger value of δ would enable fitting more BUs per territory.

The proposed algorithm is named SDR (Sandoval-Díaz-Ríos) and is shown in Algorithm 1. It uses a different version of the subproblem in each phase (namely $\text{SP}^{\text{LR}}(\delta)$, $\text{SP}^2(\delta)$, $\text{SP}^3(\delta)$, and $\text{SP}^4(\delta)$). The first phase performs a binary search over the list of distances to find an initial lower bound (LB_1 indexed by i_1^* in D) for the optimal solution of the CTDP. In the following phases the lower bound is improved to LB_2 (indexed by i_2^*) in the second phase and then to LB_3 (indexed by i_3^*) in the third phase, until a feasible solution X* to the CTDP is obtained at the end of the fourth phase. This method ensures that the solution obtained at the end of the algorithm corresponds to the optimal value (δ^*).

```
Algorithm 1: SDR Algorithm
  Input: An instance of the CTDP
  Output: An optimal solution $X^*$
  Phase I:
  $i_1^* \leftarrow binary\_search(D)$
  Phase II:
  $i_2^* \leftarrow update\_LB(\text{SP}^2(\delta), i_1^*)$
  Phase III:
  $i_3^* \leftarrow update\_LB(\text{SP}^3(\delta), i_2^*)$
  Phase IV:
  $X^* \leftarrow ensure\_connectivity(i_3^*)$
  return $X^*$
```

Phase I of the algorithm uses the linear relaxation of the subproblem $\text{SP}^{\text{LR}}(\delta)$ to perform a binary search over the list of distances. This process is described in pseudocode in Algorithm 2. The input in this phase of the algorithm is the list of distances and the output is the index i_1^* of the initial lower bound LB_1 for the objective function of the CTDP. The first step is to initialize the values of the variables for the lower and upper limits of the list ($l = 1$ and $u = K$, respectively) so that the value located at the center of the list (indexed by $\lfloor(l+u)/2\rfloor$) is the first candidate δ. The optimal value (p^*) of $\mathrm{SP}^{\mathrm{LR}}(\delta)$ is compared to p to determine the next value to be tested. Recall that if $p^* > p$ the value of δ is lower than the optimal distance value of the CTDP. Thus the next value to be tested will be the central value of the upper half of the list, which is greater than δ. Conversely, if $p^* \leq p$ the next value to be tested must be smaller so it will be the central value of the lower half of the list. (These moves are performed by changing indexes l and u to determine the index of the next value by $\lfloor(l+u)/2\rfloor$.) This process is repeated in an iterative manner until a single value from the list is left (when $l \geq u$). The δ that was tested last is the initial lower bound LB_1 for the optimal solution of the CTDP since it is the smallest distance value that enables the creation of (at most) p territories under the constraints of $\mathrm{SP}^{\mathrm{LR}}(\delta)$.

Algorithm 2: *binary_search*(D)

```
Input: $D = \{d_1, d_2, ..., d_K\}$: ordered list of all the different distance values between BUs.
Output: $i$ the index of the lower bound for the CTDP objective function.
Initialize the lower and upper indexes of the limits of the list:
$l \leftarrow 1$
$u \leftarrow K$
while $l < u$ do
    $i \leftarrow \lfloor(l+u)/2\rfloor$
    $\delta \leftarrow d_i$
    $p^* \leftarrow solveLP(\mathrm{SP}^{\mathrm{LR}}(\delta))$
    if $(p^* > p)$ or $(p = \mathrm{NULL})$ then
        Move to the upper half of the list of distances:
        $l \leftarrow i + 1$
    else
        Move to the lower half of the list of distances:
        $u \leftarrow i - 1$
    end
end
return $(i)$
```

Next, in Phase II the subproblem used changes to an IP model $\mathrm{SP}^2(\delta)$ that is obtained by adding to $\mathrm{SP}^{\mathrm{LR}}(\delta)$ the integrality constraints to the decision variables that correspond to territory centers x_{ii}. This means that $\mathrm{SP}^2(\delta)$ differs from model $\mathrm{SP}(\delta)$ by changing constraint (13) to:

$$x_{ii} \in \{0,1\} \quad \forall i \in V \tag{14}$$

$$x_{ij} \in [0,1] \quad \forall i,j \in V \quad | \quad i \neq j \tag{15}$$

The index of LB_1 is an input that is used as the δ to be tested first with $\mathrm{SP}^2(\delta)$. This time around, if the obtained p^* is different from p or the problem is infeasible (meaning $p = \mathrm{NULL}$), the next greater value from the list is tested. The process is repeated until the value of p is reached and the current candidate value is the new lower bound (LB_2) that will be tested in the next phase of the algorithm. In Phase III the subproblem $\mathrm{SP}^3(\delta)$ is an IP model (notice that $\mathrm{SP}^3(\delta) = \mathrm{SP}(\delta)$). The process in this phase is the same as in Phase II (differing only in the subproblem used) and is described in pseudocode in Algorithm 3. The current lower bound (LB_2) is tested with the $\mathrm{SP}^3(\delta)$ and the value of the lower bound is increased if necessary.

```
Algorithm 3: update_LB(model, i*)
  Input: (model, $i^*$): the model to be used and the index of a valid lower bound.
  Output: $i$ : the index of an improved lower bound.
  Initialize the index of the list at the current lower bound.
  $i \leftarrow i^*$
  $\delta \leftarrow d_{i^*}$
  continue ← TRUE
  while continue do
      $p^* \leftarrow solveIP(model(\delta))$
      if $(p^* > p)$ or $(p = \text{NULL})$ then
          Move to the next greater value in the list of distances:
          $i \leftarrow i + 1$
          $\delta \leftarrow d_i$
      else
          The current $\delta$ is an adequate LB for this problem.
          continue ← FALSE
      end
  end
  return $(i)$
```

A solution obtained in Phase III may not satisfy the connectivity constraints (6). Recall these constraints were relaxed and are not included in $\mathrm{SP}(\delta)$. The idea of Phase IV is to solve model $\mathrm{SP}^4(\delta)$, which is $\mathrm{SP}(\delta)$ plus the connectivity constraints. Clearly, given the exponential number of connectivity constraints, model $\mathrm{SP}^4(\delta)$ is solved in a similar fashion to the method by [27]. The idea is to start $\mathrm{SP}^4(\delta)$ as $\mathrm{SP}(\delta)$, then in an iterative manner identify violated cuts and add them to $\mathrm{SP}^4(\delta)$. This is depicted in Algorithm 4. As pointed out by [27], identifying and generating violated inequalities (6) is a particular separation problem that is simple since it involves finding all the connected components of each territory in a given solution, which can be done in polynomial time. If it turns out every territory is formed by a single connected component, then there are no violations and the algorithm terminates with an optimal solution to $\mathrm{SP}^4(\delta)$. Otherwise, each connected component found that does not contain a territory center gives rise to a connectivity constraint (where S is the connected component disconnected from the center in constraint (6)). The constraints are then generated and added to $\mathrm{SP}^4(\delta)$ as cuts for the next iteration of the algorithm. The procedure iterates until no disconnected subsets are found. In that case the solution is also a feasible solution to the CTDP with the final δ^* tested as the optimal value.

Algorithm 4: *ensure_connectivity*(i^*)

```
Input: $i^*$: the index of a valid lower bound.
Output: $X$ : the solution vector for the CTDP.
$i \leftarrow i^*$
$\delta \leftarrow d_i$
$continue \leftarrow$TRUE
while $continue$ do
    $(X, p^*) \leftarrow solveIP(\text{SP}^4(\delta))$
    while $(p^* > p)$ or $(p = \text{NULL}))$ do
        Move to the next greater value in the list of distances:
        $i \leftarrow i + 1$
        $\delta \leftarrow d_i$
        $(X, p^*) \leftarrow solveIP(\text{SP4}(\delta))$
    end
    $Cuts \leftarrow solve\_separation(X)$
    if $(Cuts \neq \emptyset)$ then
        $add\_cuts(Cuts)$
    else
        $continue \leftarrow$FALSE
    end
end
return $X$
```

5 Empirical Evaluation

The performance of the proposed methodology was assessed by comparing it with the one introduced by [27] which is the only exact algorithm for the CTDP known to date to the best of our knowledge. Their method begins by solving a relaxed version of the CTDP model which omits connectivity constraints and iteratively adds them as cuts by solving the previously described Separation Problem. It is a very similar process to the one performed in Phase IV of our algorithm but with the CTDP model instead of the SP model. Both methodologies were tested with instances generated based on the real case data presented by [18] from the case study of a bottled beverage firm in Monterrey, Mexico. Two types of data sets where considered, one that assumes a uniform distribution of the values of each activity measure (UD) and another that has a triangular distribution over such values (TD). Instances range in sizes of 60, 80, 100, 120, 150, 200 and 300 BUs having 20 different instances of each size per data set. From now on, the types of instances would be referred to by the prefix of the distribution of its activity measures followed by the number of nodes it contains, for example, DU60 refers to instances with a uniform distribution with 60 nodes.

The value of p is five and two activity measures are considered, namely the number of customers and product demand. The algorithms were coded in C++ using the solver from the CPLEX 12.8.0 API for Visual Studio 2015. The tests were performed in a computer with Intel(R) Xeon(R) CPU E5-2687W v2 @ 3.40GHz, a 64-bit Operating System with 8 cores and 16 logical processors. The time limit was set at 108000 seconds of CPU time for every test.

5.1 Algorithm performance

In this section we assess the quality of the lower bounds found at each phase of the proposed algorithm that will SDR1. Table 1 shows the average gaps between the lower bounds obtained in each phase and the optimal value (δ^*) for instances of the same type. It is clear that right from Phase I the lower bounds are very close to the optimal value (with gaps lower than 2%) and at the end of Phase III the gaps are nearly zero which means that almost every time the optimal value was obtained. This means that in most cases, in phase IV the only thing missing is finding a feasible assignment of BUs into territories since the optimal dispersion value has been reached already.

Table 1: Average gaps between the optimal value and the lower bounds obtained in each phase of algorithm SDR1

	Phase 1	Phase 2	Phase 3
UD60	0.01	0.01	0.00
UD80	0.01	0.00	0.00
UD100	0.02	0.00	0.00
UD120	0.01	0.00	0.00
UD150	0.01	0.00	0.00
UD200	0.01	0.00	0.00
TD60	0.01	0.00	0.00
TD80	0.01	0.00	0.00
TD100	0.01	0.00	0.00
TD120	0.01	0.00	0.00
TD150	0.01	0.00	0.00
TD200	0.02	0.00	0.00

Table 2 shows the average of both computational time and number of iterations involved in each phase of the algorithm. The number of iterations needed in Phase I grows with the size of the instance since it involves the binary search over the list of distances. The next two phases most commonly, take one or two iterations which means that the lower bound from the first phase was very close with the lower bounds from these phases. Phase IV is the one that takes greatest number of iterations in comparison with the other three because several iterations were needed to add the sufficient connectivity constraints in order to reach the optimal solution.

Table 2: Average percentage of CPU time and number of iterations in A1

	Average percentage of total CPU time				Average number of iterations			
	Phase 1	Phase 2	Phase 3	Phase 4	Phase 1	Phase 2	Phase 3	Phase 4
UD60	35.10%	18.16%	16.27%	30.15%	10.00	3.10	6.25	63.10
TD60	49.92%	21.36%	6.52%	21.74%	10.00	4.55	4.40	7.20
UD80	29.38%	35.64%	3.94%	30.89%	11.00	12.65	3.85	19.85
TD80	36.38%	26.92%	7.94%	28.53%	11.00	11.95	2.65	13.75
UD100	25.48%	38.18%	3.74%	32.47%	11.40	31.35	1.00	39.60
TD100	31.06%	29.25%	11.99%	27.55%	11.40	22.10	3.45	44.80
UD120	20.47%	40.78%	7.26%	31.41%	12.00	31.70	2.10	28.40
TD120	25.19%	19.72%	2.24%	52.75%	12.00	17.70	1.00	72.30
UD150	34.43%	34.25%	3.24%	27.96%	12.80	21.05	1.00	35.60
TD150	18.55%	45.45%	6.19%	29.74%	12.90	45.20	2.95	34.05
UD200	16.42%	29.40%	4.53%	49.60%	13.65	53.35	8.25	50.65
TD200	6.61%	64.78%	1.02%	27.56%	13.50	129.50	1.00	61.55
UD300	11.83%	48.80%	0.34%	39.01%	14.82	122.82	1.00	140.82
TD300	20.38%	58.61%	3.17%	17.79%	14.86	114.86	1.00	21.57

5.2 Comparison with previous methodology

Figures 4a to 4n show the performance comparison of the proposed algorithm SDR1 with the one presented by [27] that will from now on be referred to as SRC1 (Salazar-Ríos-Cabrera). The comparison of the algorithms is in CPU time shown in different graphs according to the type of instances tested. In most of the instances SDR1 outperforms SRC1 because it takes considerably less time. Recall that both methods are exact which means that they reach the same optimal solution, and therefore a comparison in CPU time is sufficient to compare their performance. Addressing some of the cases where SDR1 took more time than SRC1 (namely for instances: UD100-6, UD100-6 UD100-14, UD120-7, TD150-16) Phase II took a lot of time because the lower bound from the previous phase was not very tight. Instance TD80-7 is an isolated case were SDR1 performed significantly worse than SRC1. For this specific case Phase III involved several iterations to improve the previous bound and in phase IV many violations of connectivity were found (in comparison with other instances) and for this reason many iterations to add cuts to find an optimal assignment where needed.

To have a better idea of the type of violations to connectivity found in Phase IV, Figure 5 shows the histograms of the sizes of the cuts added to all instances of each type. The size of a cut corresponds to the cardinality of a connected component in a territory that does not contain the territory center. All histograms are of similar shape, most of the cuts added are of size one and they have long tails as the size of the cuts increases. The graphs show that the majority of the violations found are of size one, which is the case when a single BU is isolated from the rest of its territory.

The histograms also show that for all instance types cuts of very large sizes (close to n/p) were found. These cuts correspond to cases where the center of a territory is not connected to most of the rest of the BUs assigned to it. The cases were these cuts were found often occur in the last iterations of the algorithm. Figure 6 shows the example of instance UD60-7 where this happens. The figure shows the partial solutions from the last three iterations of the algorithm. The first image shows a territory where the center and one of its neighbors (nodes 14 and 42) are disconnected from the rest of its territory (in orange), while the second image shows a territory center (node 59) being completely disconnected from its territory (in red). Finally the last image shows the optimal solution attained in the next iteration.

It is worth mentioning that these cases are less likely to happen with the p-center since connectivity is enforced through compactness. The objective of SP is not to maximize compactness and thus it is possible to find violations of this size.

5.3 Strengthening the model

Phase IV is generally the most time-consuming since many iterations are required to find violations to connectivity and to add the correspondent cuts to reach a feasible solution. Given that many of the cuts added in the final phase of the algorithm are associated with disconnected subsets of cardinality one, one could strengthen the model by considering from the beginning all these constraints. We refer to these constraints as singleton-connectivity constraints or singleton-cuts. Note that there are a polynomial number number of these constraints. This could reduce the computational burden of Phase IV. There is an interesting trade-off when the model is strengthened this way. In one hand, it is expected that the number of iterations needed to reach an optimal solution would decrease. In the other, hand, the addition of all singleton-connectivity constraints could increase the computational time needed to solve the problem at each iteration. Therefore, one issue we investigate is precisely an assessment of this trade-off.

In the proposed algorithm, Phase IV begins by analyzing the solution of the previous phase. In the new version of the algorithm, once the first violation to connectivity is found in that solution, all singleton-cuts are added for the next iteration. This means that this time SP^4 includes all singleton-cuts for connectivity. This new algorithm from now on will be referred to as SDR2. The performance of this algorithm was compared to both the previous SDR1 and to a similar improvement to the methodology presented in [27] that will be referred to as SRC2. For SRC2, singleton-cuts are the first connectivity constraints added to the relaxed CTDP model once any violation is found.

Figures 7a to 7n display the computational (CPU) time comparison of SDR1 to SDR2. The aim was to compare the trade-off between potentially reducing the number of iterations in phase IV (since all the singleton-cuts are already added) and having to solve a problem with a higher number of constraints in each iteration due to the addition of all the singleton-cuts. The results show that there is not an evident advantage of method SDR2 over SDR1. For the instance types UD60, UD150 and TD80 the improvement is clear, specially in the cases where SDR1 performed the worst. This is related to the fact that in those instances most of the violations found with SDR1 were of size one. In the rest of the graphs the performance of SDR2 is close to that of SDR1 and in some instances, it is even more time-consuming. For example, in the cases of UD120-15 and TD120-13, the number of cuts were significantly reduced, however a large number of cuts of greater sizes were still needed to reach the optimal solution.

We conducted a Wilcoxon Signed Test (WST) to measure the difference between both methods. WST is a non-parametric test used to compare two matched samples to assess if their expected values differ [26]. For the purposes of this paper the two matched samples are the CPU time results for methods SDR1 and SDR2. The null hypothesis of the WST is then $E[SDR1] = E[SDR2]$ and the alternative is $E[SDR1] > E[SDR2]$. The results of the p-values of these tests are shown in Table 3.

Table 3: Wilcoxon Signed Test for the CPU time results between SDR1 and SDR2

Instance Type	WST p-value
UD60	0.02499
TD60	0.35860
UD80	0.66290
TD80	0.00424
UD100	0.36140
TD100	0.39210
UD120	0.37810
TD120	0.04865
UD150	0.31080
TD150	0.21520
UD200	0.93840
TD200	0.60790

P-values indicate the significance level at which the CPU times from SDR1 are greater than SDR2. Since in most cases the p-value is greater than 0.05 the null hypothesis cannot be rejected, therefore SDR2 cannot be considered an improvement to method SDR1. However, it is still advisable to use SDR2 because the results do not show either that it performed worse overall than SDR1, and it has proven to be helpful for the cases where several iterations are needed to add singleton-cuts.

Finally, we compared the performance of SDR2 with SRC2, the algorithm presented by Salazar-Aguilar et al. [27] which includes all singleton-cuts once a violation to connectivity is found. The CPU time results for these two algorithms is shown in Figures 8a to 8j. Again it is clear that for most cases the use of the sub-problems with the proposed methodology outperforms algorithm SRC. Two outlayers are instances TD80-7 and TD150-16 in which even though SDR2 performed better than SDR1, it still fails to outperform SRC2. This could be attribuited to the fact that the lower bounds in the intermediate phases (LB_2 for TD80-7 and LB_1 for TD150-16) where too far from the optimum and thus several iterations where required to reach the optimal solution.

6 Conclusions

In this paper we have presented an exact solution method for a commercial TDP under the minimization of the p-center dispersion function. The problem considers connectivity and multiple balance as planning requirements to ensure a fair and efficient allocation of workload to salesmen. The range of its objective function is defined by the set of distance values between every pair of BUs in an instance. The proposed method of solution is an iterative algorithm in which different distance values from that set are tested as candidates of the objective value in search of an optimal solution. Testing candidate distance values was possible with the definition of an auxiliary subproblem SP(δ) that minimizes the number of territories formed with the fixed value δ of the p-center dispersion.

The algorithm consists of four phases in which a different version of SP(δ) is used to test candidate distance values. The algorithm progressively improves the lower bound of the objective function of the original problem by adding constraints to the subproblem. Connectivity constraints are included until the final phase of the algorithm. They are added as cuts that correspond to violations of connectivity found in partial solutions. Once no more violations to connectivity are found, the optimal solution is reached.

The proposed algorithm, SDR1, was tested with randomly generated instances from real-case data. The results indicate that in the final phase the addition of violated connectivity cuts involved a lot of computational time and that most of the cuts added corresponded to violations of size one. With this in mind a second version, SDR2, of the algorithm was tested, this time adding all of the cuts for violations of size one at the beginning of the final phase. The performance of both versions was validated by comparing it with the methodology presented by Salazar-Aguilar et al. [27] for the same problem (with matching strategies for the addition of cuts for connectivity). Results show that the methodology proposed in this paper significantly outperforms the existing one in most cases because it takes significantly less CPU time to reach an optimal solution. Even though SDR2 cannot be considered as an improvement to SDR1 from the statistical tests, it is still advisable to use SDR2 over SDR1 since it is not diminishing neither.

Future work for the improvement of this methodology could include the implementation of a heuristic strategy to narrow the search space with upper bounds to the objective function. The method could also be improved with additional constraints to the subproblems related to either connectivity or compactness with the aim of reducing the cuts needed to reach a feasible solution. This might be helpful since both connectivity and compactness are relaxed in the objectives of the subproblems used.

Acknowledgments: The authors would like to thank both Universidad de las Americas Puebla (UDLAP) and the Mexican National Council for Science and Technology (CONACYT) for the scholarships granted to the first author.

References

1. Ahuja, N., Bender, M., Sanders, P., Schulz, C., and Wagner, A. (2015). Incorporating road networks into territory design. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, Seattle.

2. Butsch, A. (2016). *Districting Problems-New Geometrically Motivated Approaches*. PhD thesis, Karlsruher Institut für Technologie (KIT), Karlsruher, Germany.

3. Camacho-Collados, M., Liberatore, F., and Angulo, J. M. (2015). A multi-criteria police districting problem for the efficient and effective design of patrol sector. *European Journal of Operational Research*, 246(2):674–684.

4. Chen, H., Cheng, T., and Ye, X. (2019). Designing efficient and balanced police patrol districts on an urban street network. *International Journal of Geographical Information Science*, 33(2):269–290.

5. D'Amico, S. J., Wang, S.-J., Batta, R., and Rump, C. M. (2002). A simulated annealing approach to police district design. *Computers & Operations Research*, 29(6):667–684.

6. Duque, J. C., Ramos, R., and Suriñach, J. (2007). Supervised regionalization methods: A survey. *International Regional Science Review*, 30(3):195–220.

7. Elizondo-Amaya, M. G., Ríos-Mercado, R. Z., and Díaz, J. A. (2014). A dual bounding scheme for a territory design problem. *Computers & Operations Research*, 44:193–205.

8. Fernández, E., Kalcsics, J., Nickel, S., and Ríos-Mercado, R. Z. (2010). A novel maximum dispersion territory design model arising in the implementation of the WEEE-directive. *Journal of the Operational Research Society*, 61(3):503–514.

9. Gac, I., Martínez, F., and Weintraub, A. (2009). A deterministic linear optimization model for allocating schools to zones. *Journal of the Operational Research Society*, 60(7):895–905.

10. Gentry, S., Chow, E., Massie, A., and Segev, D. (2015). Gerrymandering for justice: Redistricting U.S. liver allocation. *Interfaces*, 45(5):462–480.

11. Gliesch, A., Ritt, M., and Moreira, M. C. (2018). A Multistart Alternating Tabu Search for Commercial Districting. In Liefooghe, A. and López-Ibáñez, M., editors, *Evolutionary Computation in Combinatorial Optimization*, volume 10782 of *Lecture Notes in Computer Science*, pages 158–173, Cham, Switzerland. Springer.

12. Hess, S. W. and Samuels, S. A. (1971). Experiences with a sales districting model: Criteria and implementation. *Management Science*, 18(4):P41–P54.

13. Hess, S. W., Weaver, J., Siegfeldt, H., Whelan, J., and Zitlau, P. (1965). Nonpartisan political redistricting by computer. *Operations Research*, 13(6):998–1006.

14. Hu, F., Yang, S., and Xu, W. (2014). A non-dominated sorting genetic algorithm for the location and districting planning of earthquake shelters. *International Journal of Geographical Information Science*, 28(7):1482–1501.

15. Kalcsics, J., Nickel, S., and Schröder, M. (2005). Towards a unified territorial design approachapplications, algorithms and GIS integration. *TOP*, 13(1):1–56.

16. Kalcsics, J. and Ríos-Mercado, R. Z. (2019). Districting problems. In Laporte, G., Nickel, S., and Saldanha da Gama, F., editors, *Location Science*, chapter 24. Springer, 2nd edition edition. Forthcoming.

17. Lari, I., Ricca, F., Puerto, J., and Scozzari, A. (2016). Partitioning a graph into connected components with fixed centers and optimizing cost-based objective functions or equipartition criteria. *Networks*, 67(1):69–81.

18. López-Pérez, J. F. and Ríos-Mercado, R. Z. (2013). Embotelladoras ARCA uses operations research to improve territory design plans. *Interfaces*, 43(3):209–220.

19. Özsoy, F. A. and Pınar, M. Ç. (2006). An exact algorithm for the capacitated vertex p-center problem. *Computers & Operations Research*, 33(5):1420–1436.

20. Ray, S., Lai, W., and Paschalidis, I. C. (2006). Statistical location detection with sensor networks. *IEEE Transactions on Information Theory*, 52(6):2670–2683.

21. Ricca, F., Scozzari, A., and Simeone, B. (2008). Weighted Voronoi region algorithms for political districting. *Mathematical and Computer Modelling*, 48(9-10):1468–1477.

22. Ricca, F., Scozzari, A., and Simeone, B. (2013). Political districting: from classical models to recent approaches. *Annals of Operations Research*, 204(1):271–299.

23. Ríos-Mercado, R. Z. and Escalante, H. J. (2016). GRASP with path relinking for commercial districting. 44:102–113.

24. Ríos-Mercado, R. Z. and Fernández, E. (2009). A reactive GRASP for a commercial territory design problem with multiple balancing requirements. *Computers & Operations Research*, 36(3):755–776.

25. Ríos-Mercado, R. Z. and López-Pérez, J. F. (2013). Commercial territory design planning with realignment and disjoint assignment requirements. *Omega*, 41(3):525–535.

26. Rosner, B., Glynn, R. J., and Lee, M.-L. T. (2006). The Wilcoxon signed rank test for paired comparisons of clustered data. *Biometrics*, 62(1):185–192.

27. Salazar-Aguilar, M. A., Ríos-Mercado, R. Z., and Cabrera-Ríos, M. (2011). New models for commercial territory design. *Networks and Spatial Economics*, 11(3):487–507.

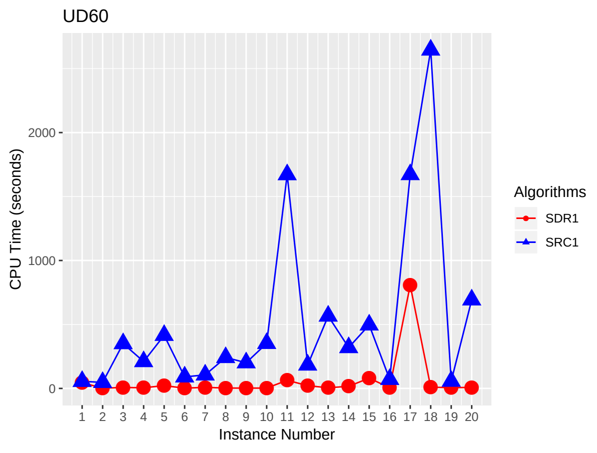
28. Salazar-Aguilar, M. A., Ríos-Mercado, R. Z., González-Velarde, J. L., and Molina, J. (2012). Multiobjective scatter search for a commercial territory design problem. *Annals of Operations Research*, 199(1):343–360.

29. Shirabe, T. (2005). Classification of spatial properties for spatial allocation modeling. *GeoInformatica*, 9(3):269–287.

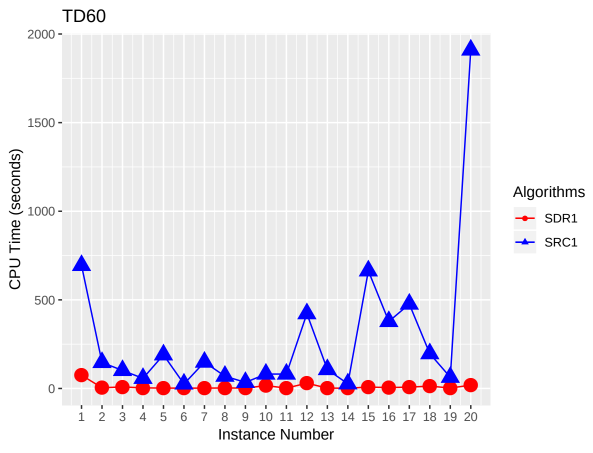
30. Shirabe, T. (2009). Districting modeling with exact contiguity constraints. *Environment and Planning B: Planning and Design*, 36(6):1053–1066.

31. Yao, G., Bi, J., Li, Y., and Guo, L. (2014). On the capacitated controller placement problem in software defined networks. *IEEE Communications Letters*, 18(8):1339–1342.

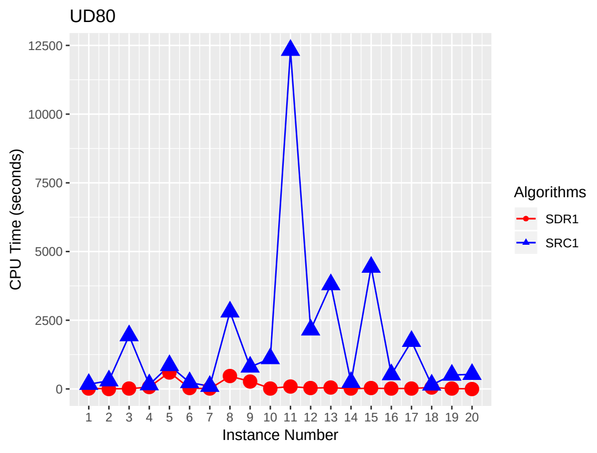
32. Zoltners, A. A. and Sinha, P. (2005). Sales territory design: Thirty years of modeling and implementation. *Marketing Science*, 24(3):313–331.



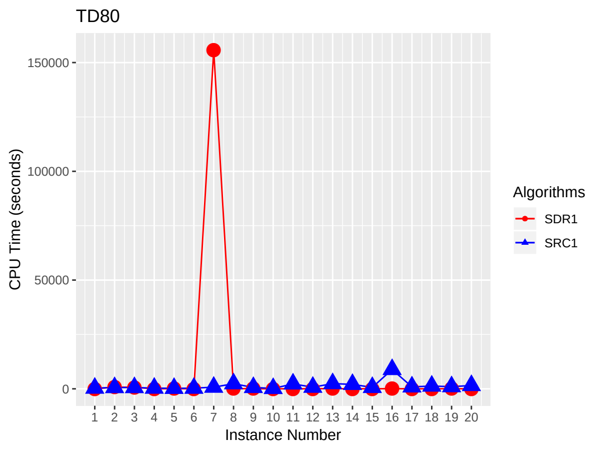
(a) UD 60



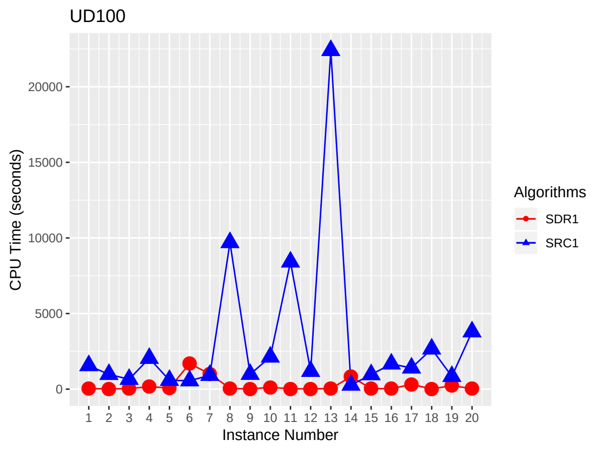
(b) TD 60



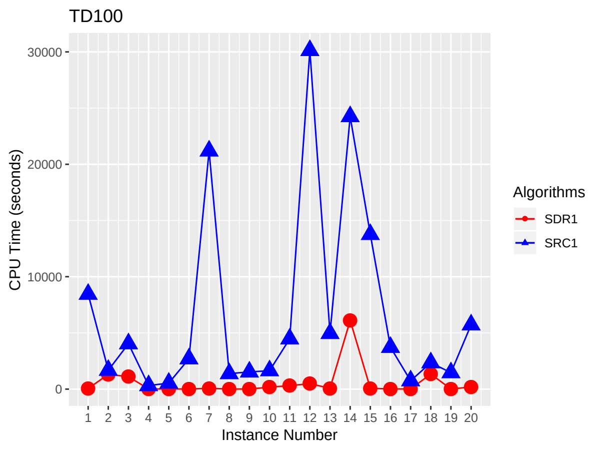
(c) UD 80



(d) TD 80

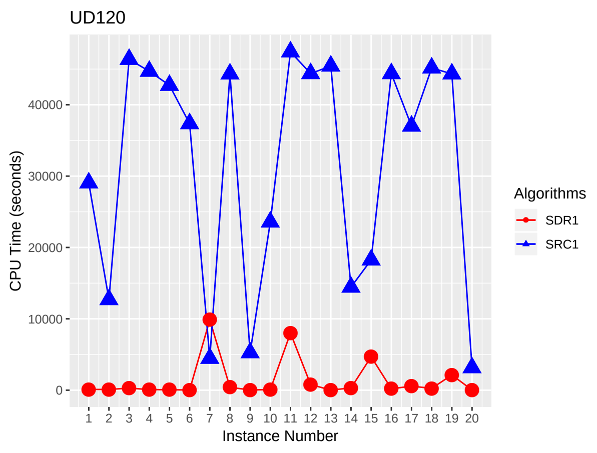


(e) UD 100

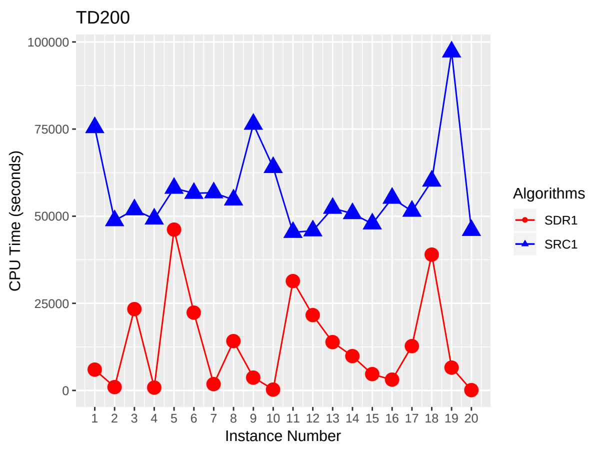


(f) TD 100

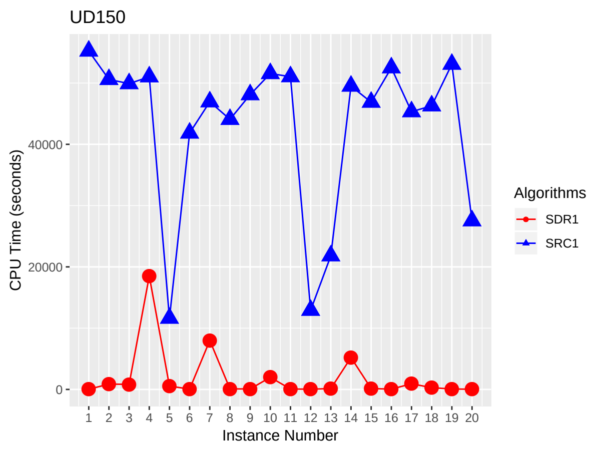
Figure 4: CPU time differences between algorithms SDR1 (in red) and SRC1 (in blue)



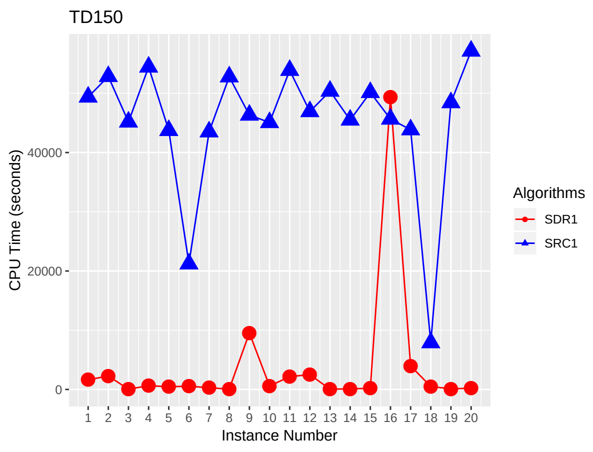
(g) UD 120



(h) TD 120

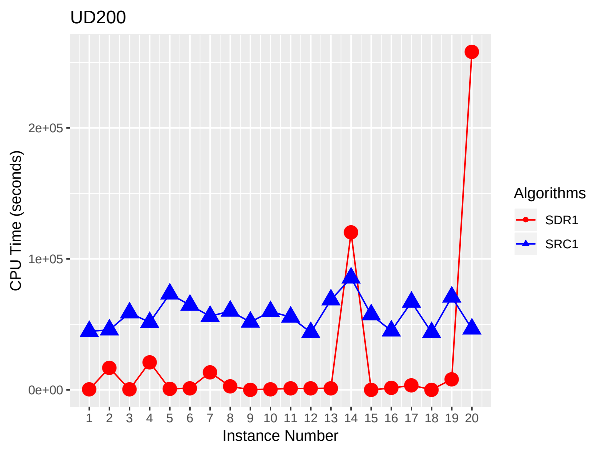


(i) UD 150

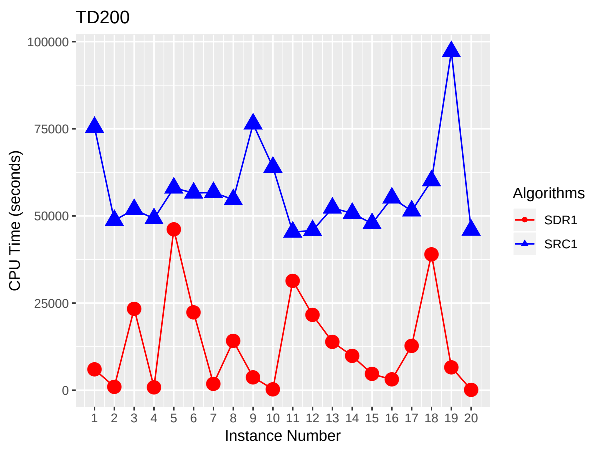


(j) TD 150

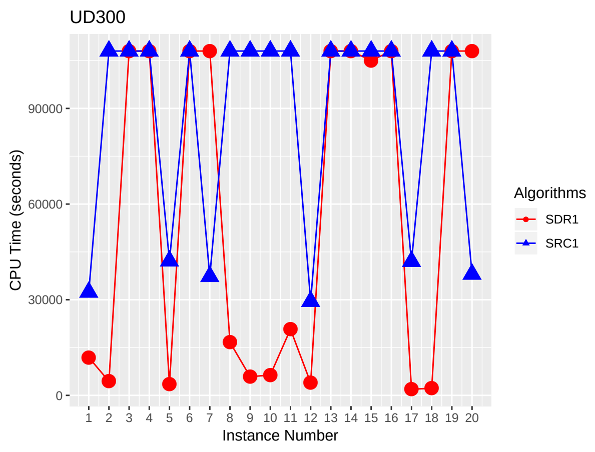
Figure 4: CPU time differences between algorithms SDR1 (in red) and SRC1 (in blue)



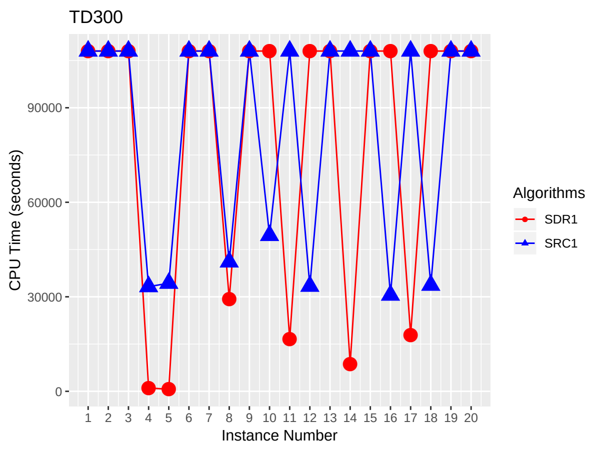
(k) UD 200



(l) TD 200

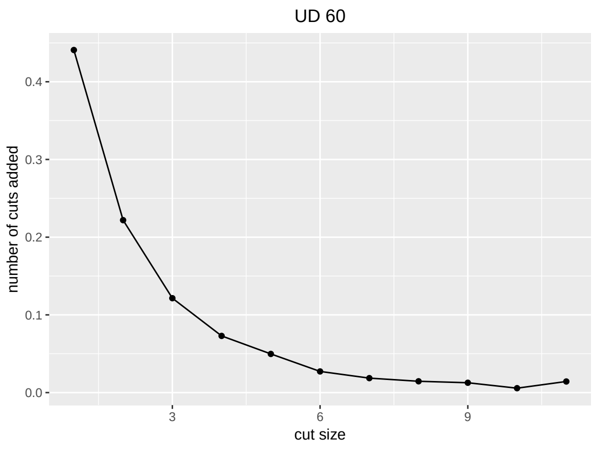


(m) UD 300

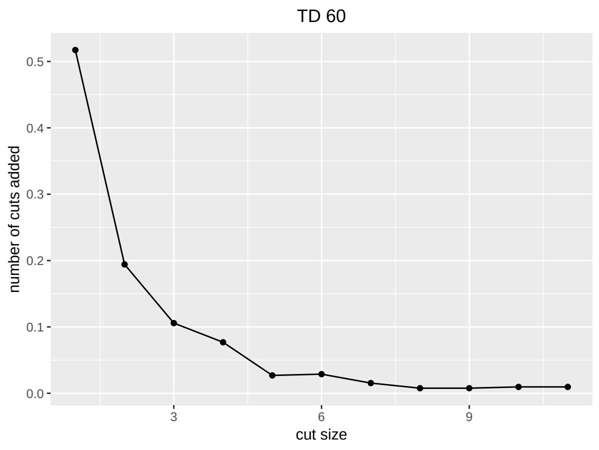


(n) TD 300

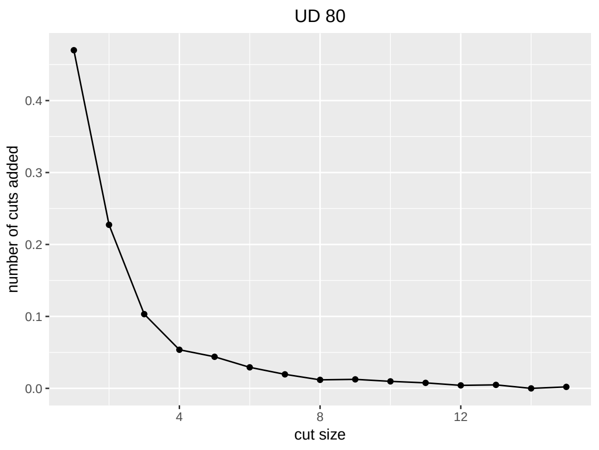
Figure 4: CPU time differences between algorithms SDR1 (in red) and SRC1 (in blue)



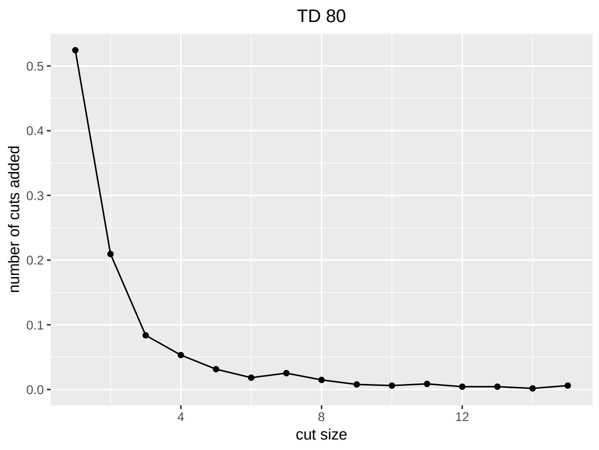
(a) UD 60



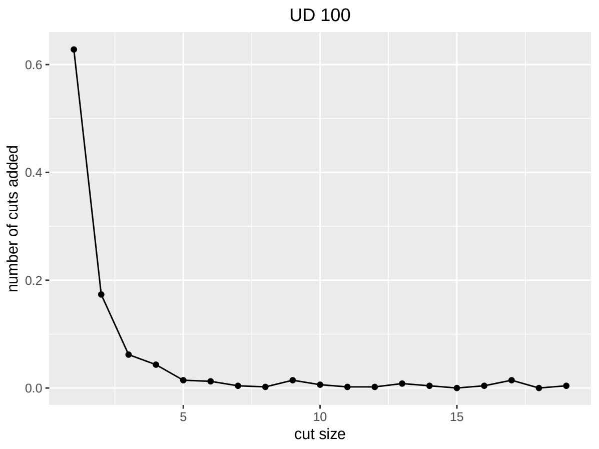
(b) TD 60



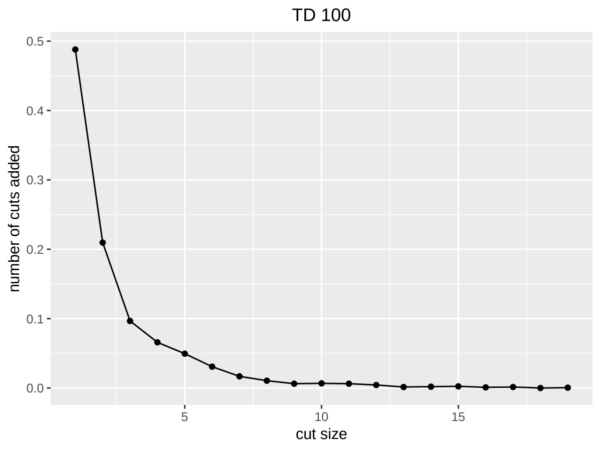
(c) UD 80



(d) TD 80

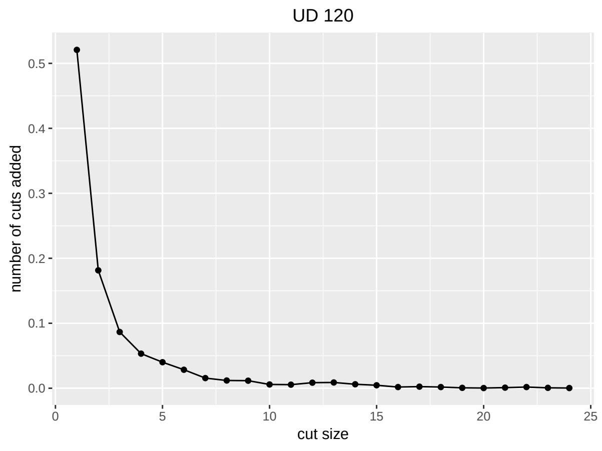


(e) UD 100

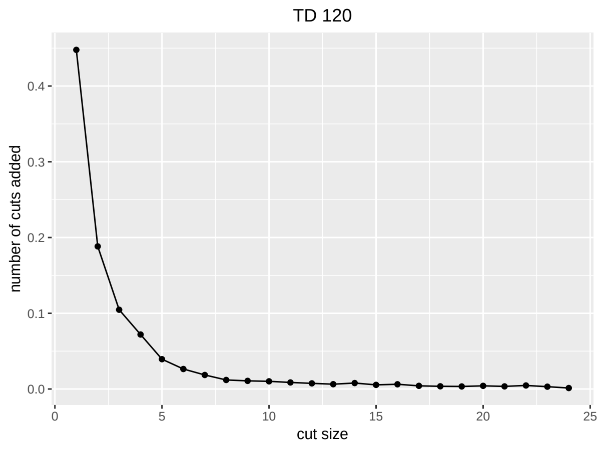


(f) TD 100

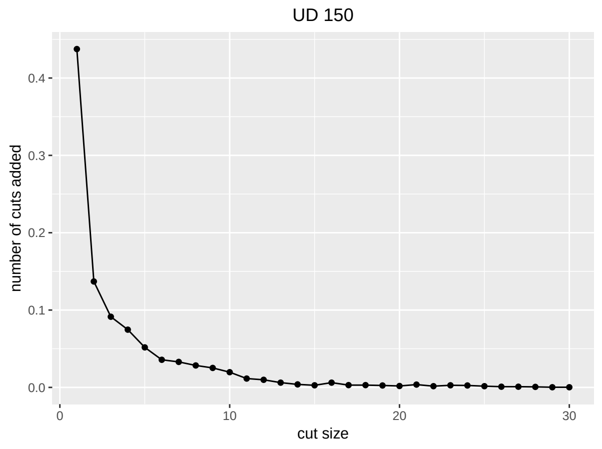
Figure 5: Histograms of the sizes of the cuts added in A1 for each instance type.



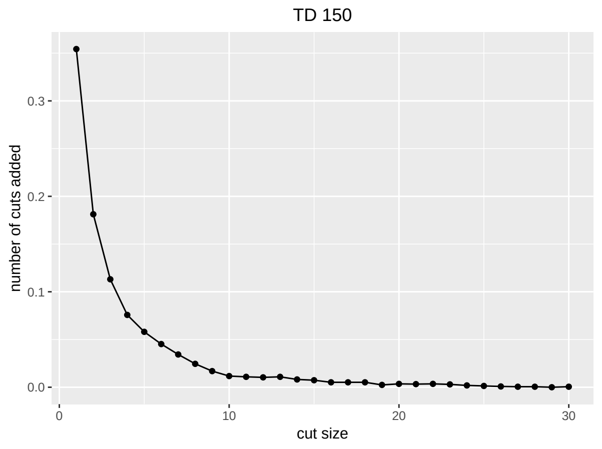
(g) UD 120



(h) TD 120

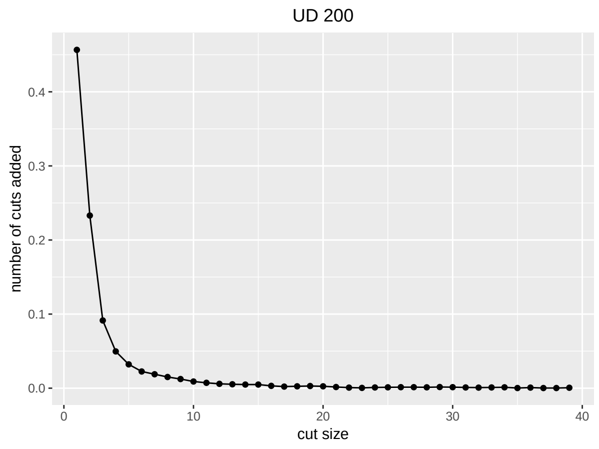


(i) UD 150

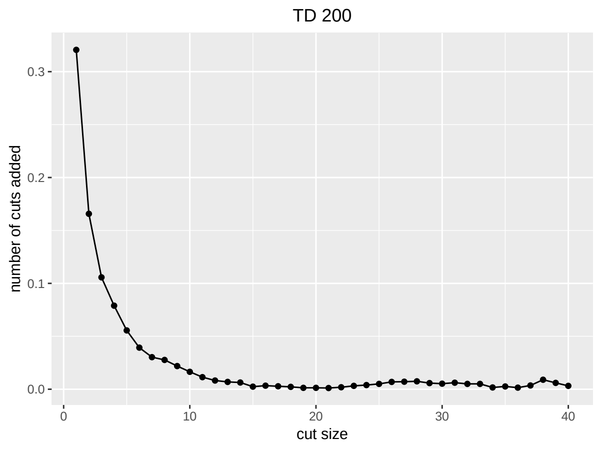


(j) TD 150

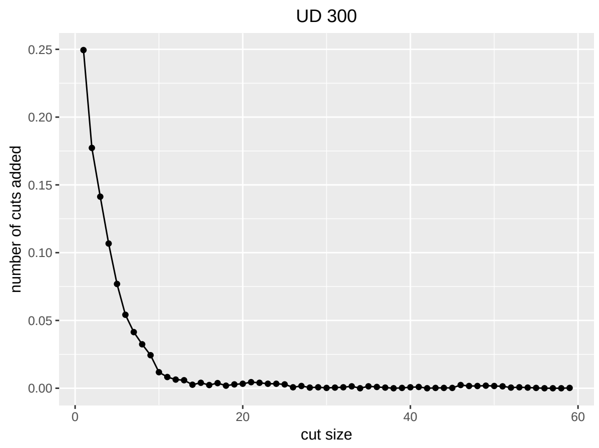
Figure 5: Histograms of the sizes of the cuts added in SDR1 for each instance type.



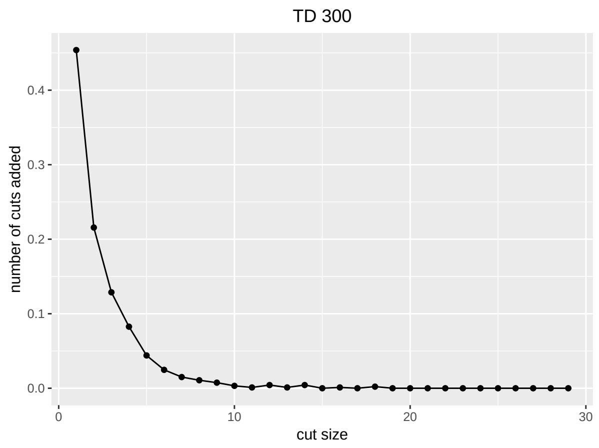
(k) UD 200



(l) TD 200

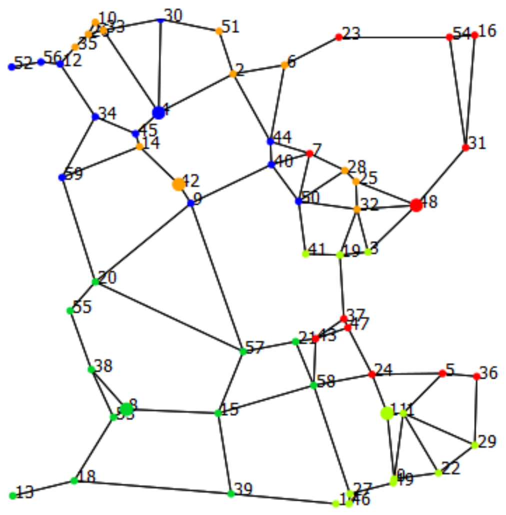


(m) UD 300

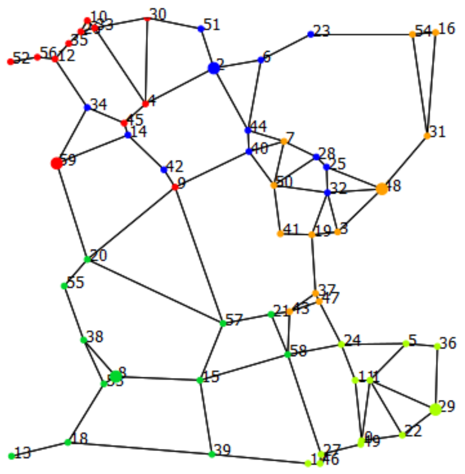


(n) TD 300

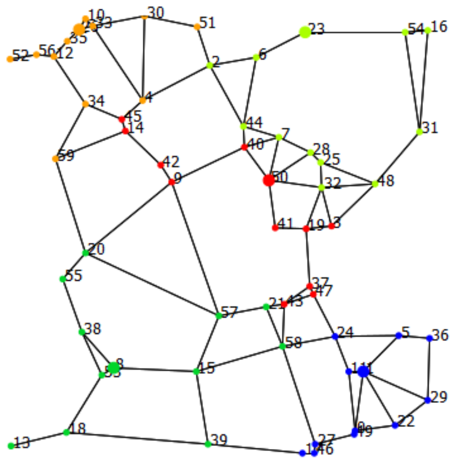
Figure 5: Histograms of the sizes of the cuts added in SDR1 for each instance type.



(a) Second-last Iteration

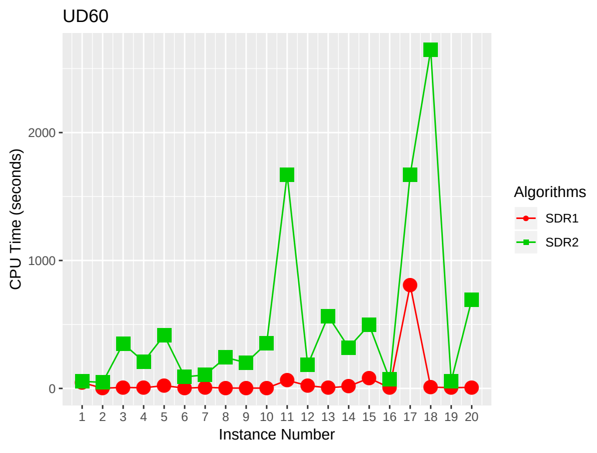


(b) Last Iteration

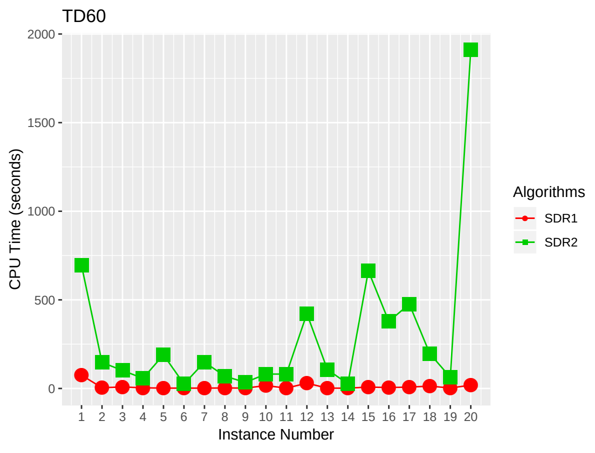


(c) Optimal Solution

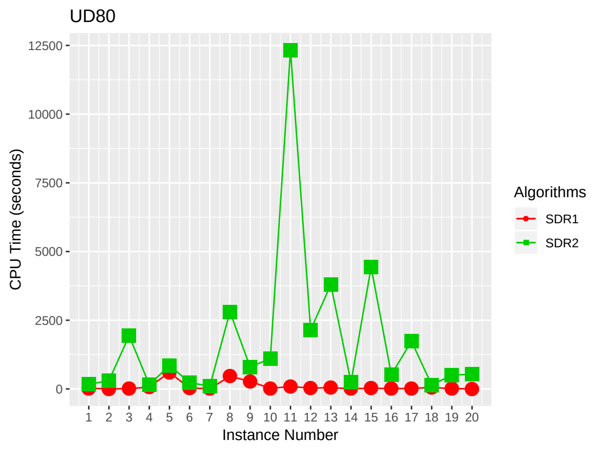
Figure 6: Partial solutions of the last three iterations of instance UD60-7



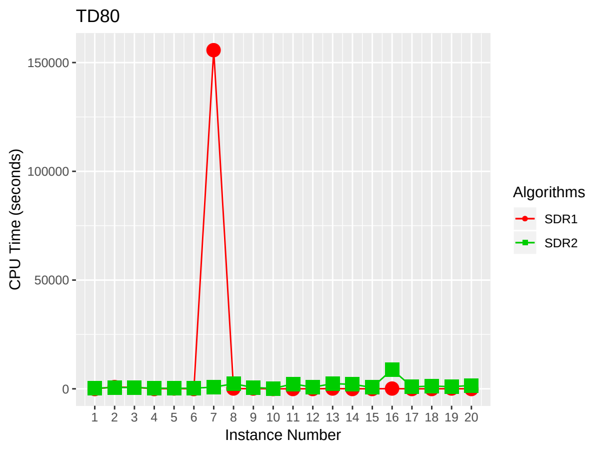
(a) UD 60



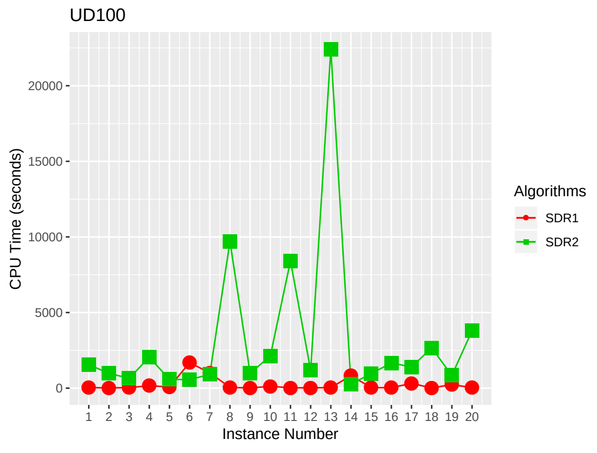
(b) TD 60



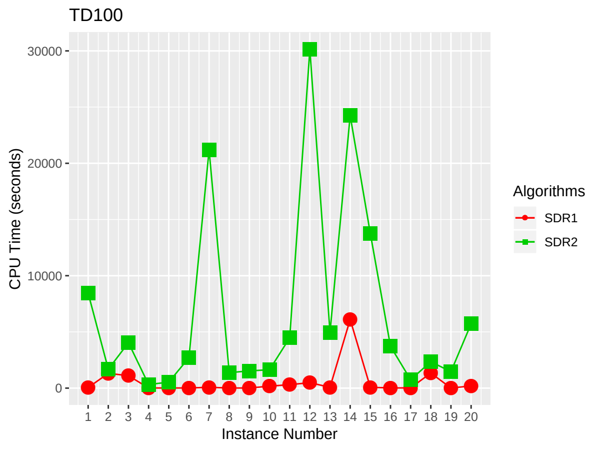
(c) UD 80



(d) TD 80

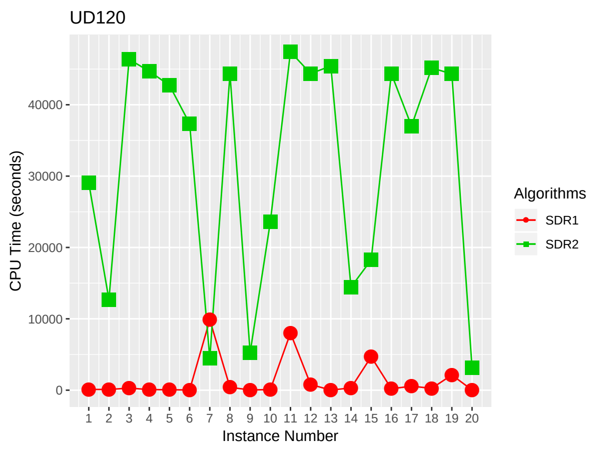


(e) UD 100

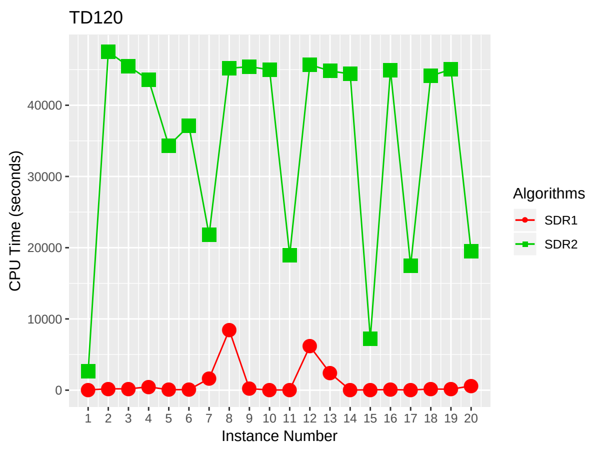


(f) TD 100

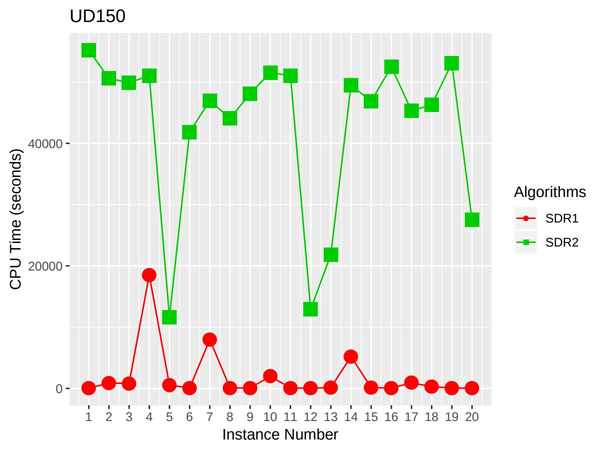
Figure 7: CPU time differences between algorithms SDR1 (in red) and SDR2 (in green)



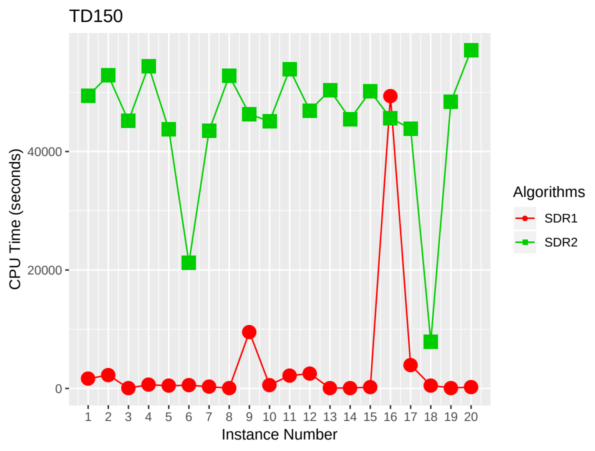
(g) UD 120



(h) TD 120

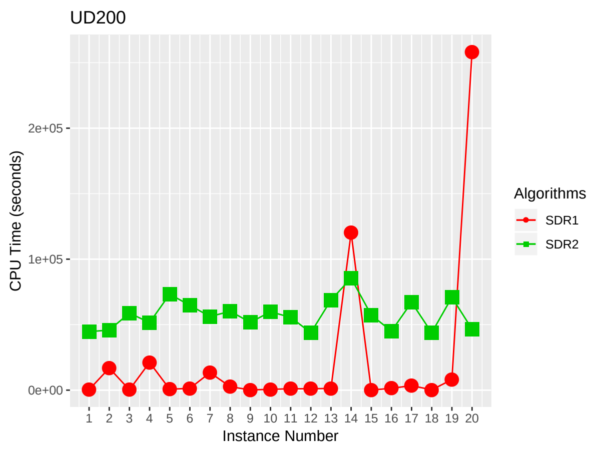


(i) UD 150

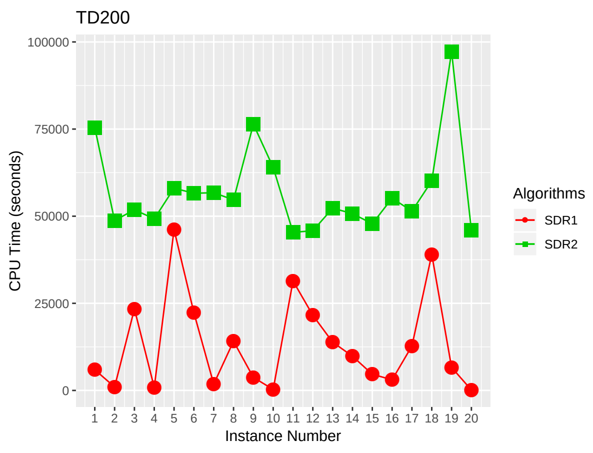


(j) TD 150

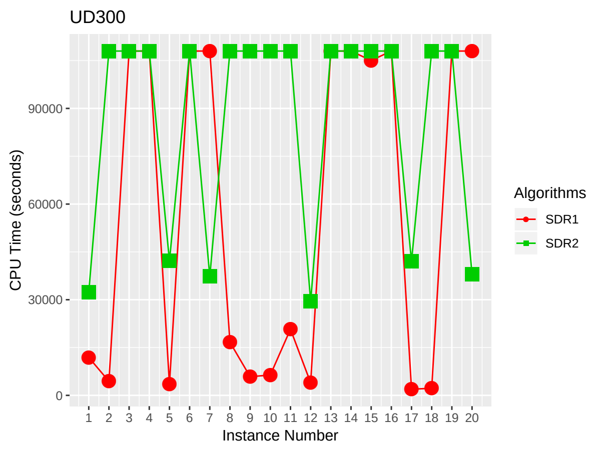
Figure 7: CPU time differences between algorithms SDR1 (in red) and SDR2 (in green)



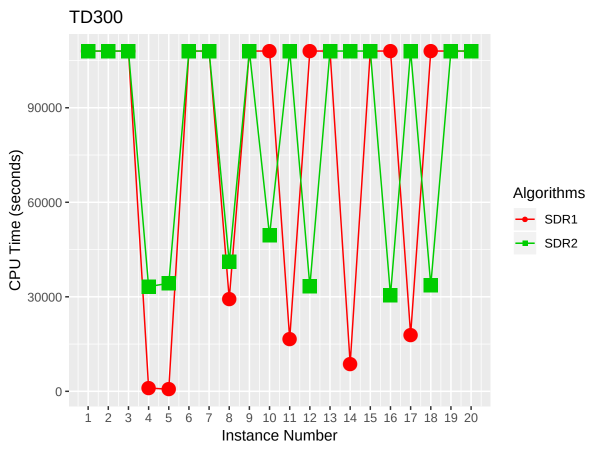
(k) UD 200



(l) TD 200

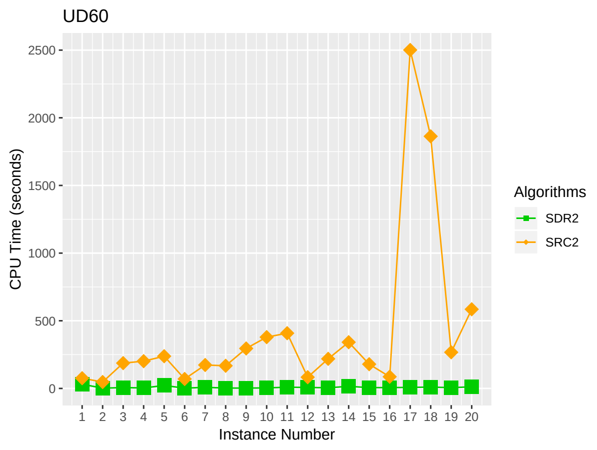


(m) UD 300

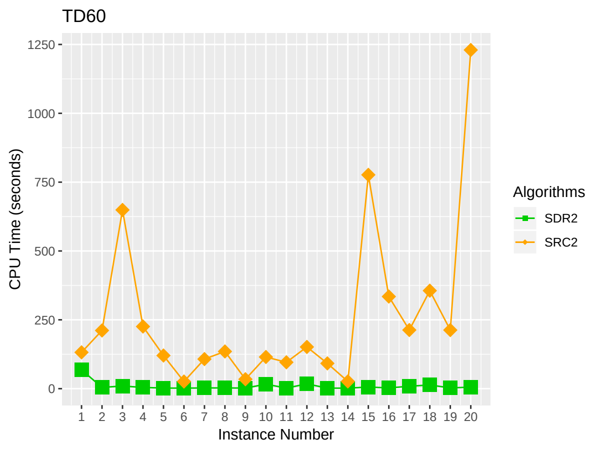


(n) TD 300

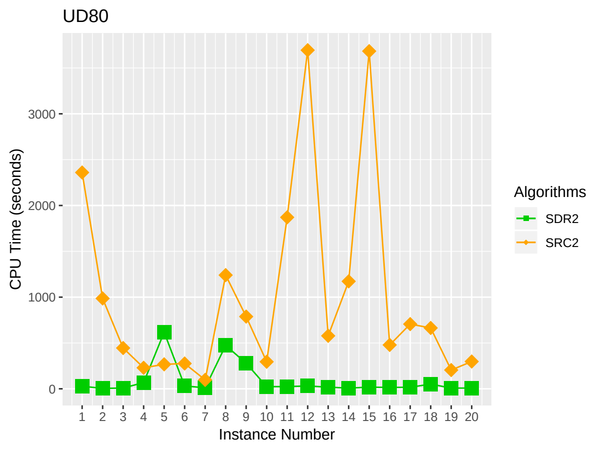
Figure 7: CPU time differences between algorithms SDR1 (in red) and SDR2 (in green)



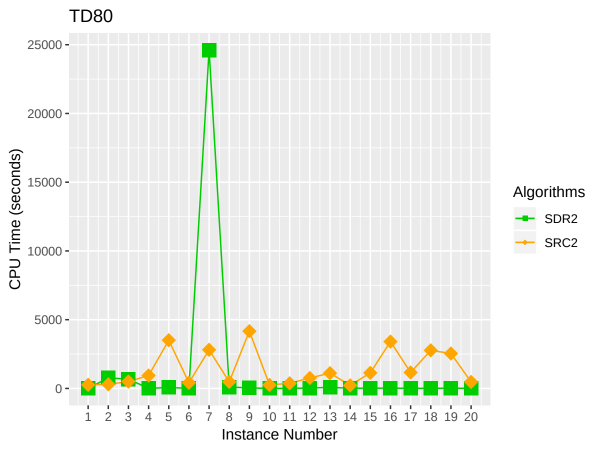
(a) UD 60



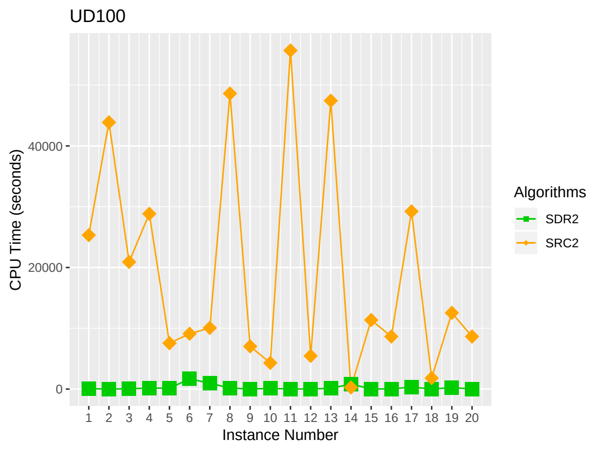
(b) TD 60



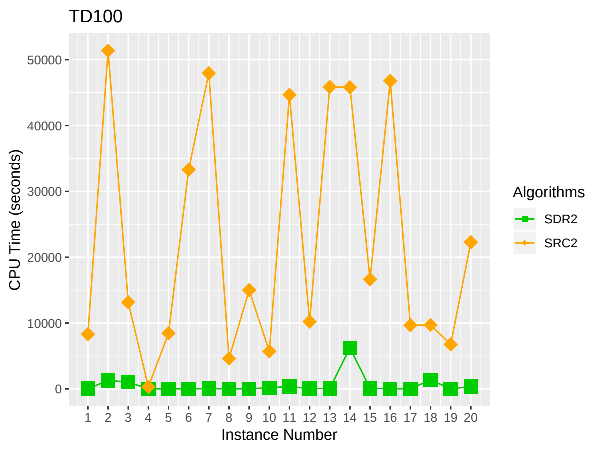
(c) UD 80



(d) TD 80

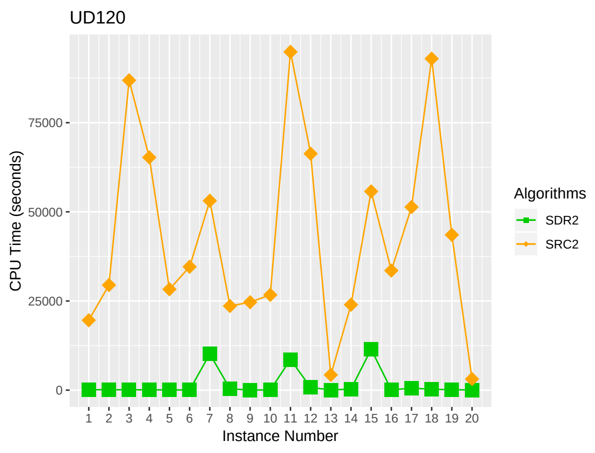


(e) UD 100

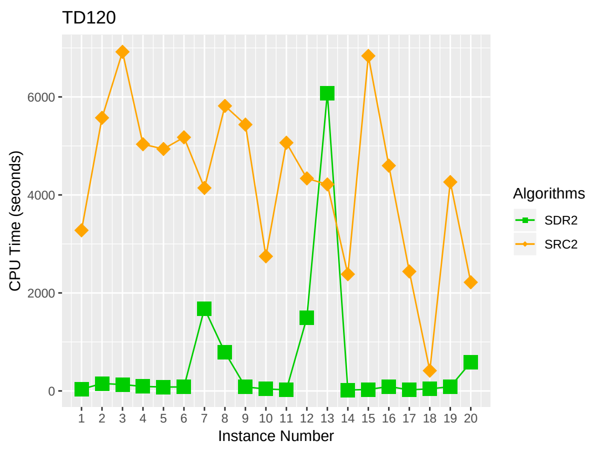


(f) TD 100

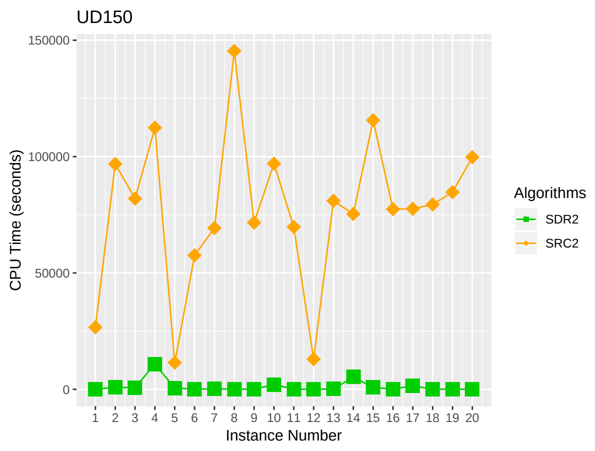
Figure 8: CPU time differences between algorithms SDR2 (in green) and SRC2 (in orange)



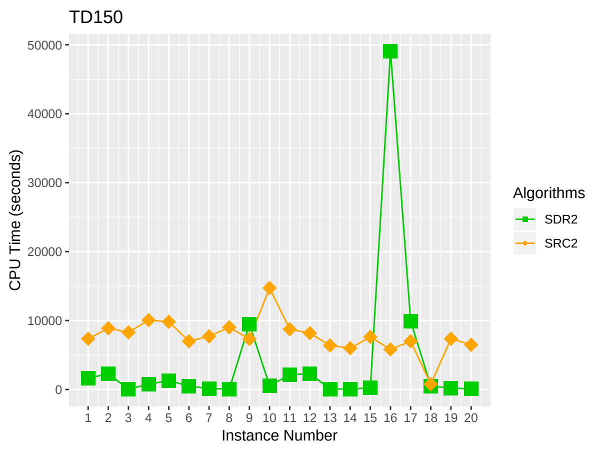
(g) UD 120



(h) TD 120



(i) UD 150



(j) TD 150

Figure 8: CPU time differences between algorithms SDR2 (in green) and SRC2 (in orange)