THE DISCRETE ONE-ROUND VORONOI GAME: OPTIMIZING THE FIRST PLAYER STRATEGY BY INTEGER PROGRAMMING

Marcos Costa Roboredo
Production Engineering Department - Fluminense Federal University
Rua Passo da Pátria 156, 24210-240, Niterói, RJ, Brazil
mcr.marcos@yahoo.com.br

Artur Alves Pessoa
Production Engineering Department - Fluminense Federal University
Rua Passo da Pátria 156, 24210-240, Niterói, RJ, Brazil
artur@producao.uff.br

RESUMO

O jogo do Voronoi discreto é um modelo matemático para o problema de localização de facildades competitiva com dois jogadores. Cada jogador deve localizar um total de p facilidades sobre um grafo alternadamente. Cada vértice do grafo é um cliente ou um candidato a facilidade de um ou dos dois jogadores. Cada cliente é dominado pelo jogador que localizar a facilidade mais próxima a este gerando um lucro para o jogador que o domina. O objetivo de cada jogador é obter o maior lucro possível. Neste artigo, consideramos o problema de otimizar a estratégia do primeiro jogador numa versão do jogo com uma única rodada, onde o cada jogador, alternadamente, escolhe onde instalar suas p facilidades de uma vez. Apesar desse problema ser $\sum_2 P$-difícil, apresentamos uma formulação por programação inteira com um número exponencial de restrições. Nós reportamos experimentos com 100 clientes e diferentes valores para p. Resultados mostram que o nosso método consome menos tempo computacional do que o consumido pelo melhor método exato encontrado na literatura, sendo capaz de resolver 10 instâncias que estavam em aberto.

PALAVRAS CHAVE. Voronoi Game, Localização de Facilidades Competitiva, Teoria dos Jogos.
Área Principal: Otimização

ABSTRACT

The discrete Voronoi game is a mathematical model for the competitive facility location problem with two players where each one must place a total of p facilities on a graph. Each node on the graph is an applicant facility of either player or a customer. Each customer is dominated by the player who owns the nearest placed facility generating a profit for the player who dominates it. Each player aims to obtain the maximum profit. In this paper, we consider the problem of optimizing the first player strategy in the one-round version of the game, where each player places its p facilities at once. Although that problem is $\sum_2 P$-hard, we present an integer programming formulation with exponentially many constraints. Moreover, we report experiments using 30 instances with 100 customers and different values of p. Results show that our method requires less computational time than the best exact algorithm found in the literature, being able to optimally solve 10 previously open instances.

KEY WORDS. Voronoi game, Competitive Facility Location, Game Theory.
Main area: Optimization

1 Introduction

The Voronoi game is a geometric model for the competitive facility location problem. That game is composed of two players (white and black) and a region U called playing arena. Each point on the playing arena is an applicant facility of either player or a customer. For each customer and each applicant (white's or black's) facility, it is defined the distance between them. Each player has to choose p among its applicant facilities to place on the playing arena one at a time alternately. Like in chess, white plays first. Each customer is dominated by the closest placed facility generating a profit for the player who dominates it. Each player aims to obtain the maximum possible profit. The player who obtains the largest profit wins. When the black starts playing after all white's p facilities are placed, the game is called the one-round Voronoi game. Otherwise, we refer to it as the classical version of the game. Two types of arena can be considered in the Voronoi game: discrete or continuous.

Many researchers have already addressed the continuous version of the Voronoi game. Ahn *et al*. (2004) presented a winning strategy (strategy where the winner dominates more customers than its opponent) for the black considering the classical version of the game and two types of arena: a circle or a line segment, but the white can keep the winning margin arbitrarily small. Cheong *et al*. (2004) also presented a winning strategy for the black in the one-round case, where the arena is a square. Fekete e Meijer (2005) solved some open questions proposed by Cheong *et al*. (2004), showing the winner for different values of p and θ (θ is a aspect ratio of the square).

Teramoto *et al*. (2006) described the discrete Voronoi game for the first time. In this game version, distances are defined either on a graph or by a distance matrix (which corresponds to a complete graph). Important results were showed by that paper: the version of the game where the players place their p facilities alternately until the moment it is not possible anymore is PSPACE-complete; for the one-round case, the decision problem of determining whether the black has a winning strategy (after the white has finished its turn) is NP-Complete; there is an optimal strategy where the white either wins or ties the game when the graph is a k-ary tree. The existence or not of a Nash equilibrium on the discrete Voronoi game has been an important research topic. Durr e Thang (2007) showed that the existence of a Nash equilibrium for general graphs is NP-hard while Mavronicolas *et al*. (2008) proposed necessary and sufficient existence criteria and exact prices of anarchy and stability on cycle graphs.

Recently, Noltemeier *et al*. (2007) proved that the general problem of finding an optimal strategy for the white (a strategy where the white dominates the maximum possible number of customers assuming that the black uses an optimal strategy) is $\sum_2 P$-hard for the discrete one-round Voronoi game. In this paper, we focus on this problem, which turns out to be harder than any optimization problem whose decision version is in NP. Let us refer to it and the White's Strategy on the Discrete One-round Voronoi Game (WSD1VG). The hardness of the WSD1VG comes from the fact that one should solve an NP-hard optimization problem to optimize the black's strategy only to evaluate a single white's solution. Fortunately, optimizing the black's strategy often spends less than one second for instances with 100 customers by solving an IP model.

From the practical point of view, we found only a few heuristics for the WSD1VG problem in the literature, the best of which being proposed by Alekseeva *et al*. (2010). The authors also propose an exact method that solved instances with up to 100 customers for $p = 5$. Although most instances were solved in a few hours, the most difficult one required more than two days of computation. Moreover, the authors could not solve instances for $p = 10$.

In this paper, we propose an integer programming (IP) formulation for the WSD1VG problem with polynomially many variables and exponentially many constraints. One should note that, since the problem is $\sum_2 P$-hard, neither a polynomial formulation nor a formulation where all constraints can be separated in polynomial time is possible unless $P = NP$. Hence, the constraints

are separated during the optimization either using a greedy heuristic or solving an IP model for an NP-hard problem. We test our method on the same 30 instances with 100 customers as Alekseeva *et al*. (2010). Our method is faster than the previous one for 18 out of 20 instances with $p = 5$, and allows for optimally solving the 10 open instances with $p = 10$.

This paper is divided as follows. In Section 2, we define the discrete one-round Voronoi Game problem and present some examples. In Section 3, we describe an IP model for the WSD1VG problem and prove that the strengthened inequalities used are valid. In section 4, we define the separation problem for the only exponential family of constraints used in our formulation, and present an IP model and a greedy heuristic for it. In section 5, we report our experiments and compare our method with the previous exact one.

2 The discrete one-round Voronoi game

The discrete one-round Voronoi game is formally defined as follows: Consider two players (White and Black). Each one of them has to place p facilities on an arena once. The second player starts playing after all p facilities of the first player are placed. As in chess, the white plays first. The game arena is a complete bipartite graph $G = (V, E)$ and each vertex $v \in V$ is either a customer or an applicant facility of the white or the black. As a result, V can be partitioned into two disjoint subsets I and J, where I is the set of applicant facilities, and J is the set of customers. The edge set E of G has an edge $e = (i, j)$ for each $i \in I$ and $j \in J$, with an associated distance d_{ij}. We say that a facility placed in a location $i \in I$ dominates a customer $j \in J$ when $d_{kj} \geq d_{ij}$ for all locations $k \in I$ that contain facilities. Each customer generates a profit w_j for the player who placed the nearest facility. Ties are broken in favor of the white's facilities, and ties between facilities of the same player are broken arbitrarily. Each player aims to obtain the maximum profit.

To illustrate the game, we propose three examples where the profit w_j for each $j \in J$ is one. The first one has $|I| = 6$, $|J| = 5$ and $p = 2$. Based on distances between customers and facilities, we order the facilities for each customer j in a non-decreasing order by distance. As a result, each row of the following matrix P gives a sorted list of all facilities for a given customer, that is, the cell P_{ji} indicates the i-th facility closest to the customer j.

$$P = \begin{bmatrix} 2 & 4 & 5 & 1 & 3 & 6 \\ 6 & 1 & 5 & 2 & 4 & 3 \\ 3 & 4 & 2 & 5 & 1 & 6 \\ 1 & 4 & 3 & 2 & 6 & 5 \\ 1 & 6 & 3 & 2 & 4 & 5 \end{bmatrix}$$

For this example, the nearest facility to the customer 2 is 6, followed by 1, and then by 5, and so on. The facility 3 is the farthest one. Suppose that the white uses the facilities 2 and 3 while the black uses the facilities 4 and 5. In this case, the customers 2 and 4 are dominated by the black and the white is the game winner, which dominates the customers 1, 3 and 5.

The second and third examples have interesting properties. For the second example, the black has a winning strategy regardless of the white's play, while in the third one, the white can always win. Both these examples have $|I| = 6$, $|J| = 6$ and $p = 2$.

The matrix P for the example 2 is the next one.

$$P = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 & 1 \\ 3 & 4 & 5 & 6 & 1 & 2 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 5 & 6 & 1 & 2 & 3 & 4 \\ 6 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

Note that, any white's strategy is a loosing one. Suppose, for example, that the white uses the facilities 2 and 4. Using the facilities 1 and 3, the black wins by dominating the customers 1,3,5 and 6. The winning strategy for the black is the following. If the white uses the facility $l > 1$ then the black should use the facility $l - 1$. Otherwise, if the white uses the facility 1, the black should use the facility 6.

The matrix P for the example 3 is the next one.

$$P = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 1 & 3 & 5 & 6 & 4 \\ 3 & 1 & 2 & 6 & 5 & 4 \\ 4 & 5 & 6 & 1 & 2 & 3 \\ 5 & 4 & 3 & 6 & 2 & 1 \\ 6 & 4 & 2 & 3 & 1 & 5 \end{bmatrix}$$

For this example, the white wins by using the facilities 1 and 4. Suppose that it occurs and the black uses the facilities 2 and 6, for example. Note that, in this case, the white dominates the customers 1,4,3 and 5. In general, the white has a winning strategy because the first two nearest facilities nearer to all customers include either the facility 1 or the facility 4.

3 The model

In this section, we show that, besides its complexity, the WSD1VG problem admits an IP formulation with polynomially many variables and exponentially many constraints. Let the binary variables x_i indicate whether the white places a facility at the location i and the binary variables y_{ij} indicate whether the white's facility nearest to the customer j is placed at the location i. Finally, let the integer variable z give the total black's profit. Let S be the set of strategies for any of the two players. It means that each $S_0 \in S$ is a set of p facilities placed by black or white. The complete model (I) is below.

$$\begin{aligned}
\min \quad & z & & (1)\\
\text{subject to} \quad & \sum_{i \in I} x_i = p & & (2)\\
& y_{ij} \leq x_i & , \forall j \in J, \forall i \in I. & (3)\\
& \sum_{i \in I} y_{ij} = 1 & , \forall j \in J. & (4)\\
z \geq & \sum_{j \in J} w_j - \sum_{j \in J} \sum_{i \in I | d_{ij} \leq \min\{d_{kj} | k \in S_0\}} w_j y_{ij} & , \forall S_0 \in S. & (5)\\
& y_{ij} \in \{0,1\} & , \forall i \in I, \forall j \in J. & (6)\\
& x_i \in \{0,1\} & , \forall i \in I. & (7)
\end{aligned}$$

The objective function (1) minimizes the number of customers dominated by the black. The first set of constraints (2) indicates that the white has to place exactly p facilities. Constraints (3) ensure the consistency between the variables x_i and the y_{ij}. The set of constraints (4) indicates that for each customer j, exactly one facility i is the white's facility nearest to j. Finally, (5) ensure, for each possible strategy $S_0 \in S$ for the black, the total black's profit z is not smaller than total sum of profits $\sum_{j \in J} w_j$ minus the number of customers that are nearest to a white's facility than to any location $k \in S_0$. Note that (5) is composed of an exponential number of constraints since S is composed of $\binom{|I|}{p}$ strategies. Hence, it is necessary to solve the separation problem associated to (5) in order to include in the formulation only the necessary constraints.

Now, in order to improve the formulation (I), we propose a family of strengthened valid inequalities by lifting (5). The main weakness of the previous family of inequalities is the fact that the lower bound on z becomes zero if the white uses the same strategy S_0 that defines the inequality because of the tie breaking criterion. As a result, fractional solutions for the linear relaxation of (I) tend to weaken the obtained lower bound on z by using convex combinations several good strategies of S. The proposed family of inequalities try to overcome this difficulty by considering other strategies for the black, in addition to S_0, when computing the lower bound on z, for the cases where the white places at least one facility that belongs to S_0. For that, we define a function $H : S_0 \rightarrow I$ that gives an alternative place for each facility placed by the black to be used if the original place has already been used by the white. Then, in addition to the black's strategy S_0, the lower bound on z also considers the strategies that replace some facilities $i \in S_0$ by $H(i)$. The new family of inequalities is the following:

$$z \geq \sum_{j \in J} w_j - \sum_{j \in J} \sum_{\substack{i \in I | ((d_{ij} \leq \min\{d_{kj} | k \in S_0\} \wedge i \notin S_0) \\ \vee \\ (d_{ij} \leq \min\{\min\{d_{kj} | k \in S_0\}, d_{H(i)j}\} \wedge i \in S_0))}} w_j y_{ij}. \quad \forall S_0 \in S, \forall H : S_0 \rightarrow I \qquad (8)$$

Theorem 1 proves the lifted inequalities (8) are valid.

Theorem 1 *The lifted cuts* (8) *are valid.*

Proof: To prove this theorem, we show that, for any white's strategy S_w, there exists a black's strategy S_b such that the associated black's profit is at least the right-hand side of (8). For that, let $\bar{y}$ be the 0-1 incidence vector for the y variables that correspondes to S_w, that is, $\bar{y}_{ij}$ is equal to one if $i = \arg\min_{k \in S_w}\{d_{kj}\}$, and equal to zero otherwise. Also, let us define the black's strategy S_b in this way:

$$S_b = S_0 \backslash S_w \cup \{H(k) | k \in S_w \cap S_0\}.$$

We must show that

$$\sum_{j \in J} w_j - \sum_{j \in J} \sum_{i \in I | d_{ij} \leq \min\{d_{kj} | k \in S_b\}} w_j \bar{y}_{ij} \geq$$

$$\geq \sum_{j \in J} w_j - \sum_{j \in J} \sum_{\substack{i \in I | ((d_{ij} \leq \min\{d_{kj} | k \in S_0\} \wedge i \notin S_0) \\ \vee \\ (d_{ij} \leq \min\{\min\{d_{kj} | k \in S_0\}, d_{H(i)j}\} \wedge i \in S_0))}} w_j \bar{y}_{ij}, \qquad (9)$$

or equivalently,

$$\sum_{j \in J} \sum_{i \in I | d_{ij} \leq \min\{d_{kj} | k \in S_b\}} w_j \bar{y}_{ij} \leq \sum_{j \in J} \sum_{\substack{i \in I | ((d_{ij} \leq \min\{d_{kj} | k \in S_0\} \wedge i \notin S_0) \\ \vee \\ (d_{ij} \leq \min\{\min\{d_{kj} | k \in S_0\}, d_{H(i)j}\} \wedge i \in S_0))}} w_j \bar{y}_{ij}, \qquad (10)$$

Now, let us define the following two sets respectively related to the left and the right-hand side of (10):

$$\overline{L} = \{(i, j) \in I \times J | d_{ij} \leq \min\{d_{kj} | k \in S_b\}.$$

$$\overline{R} = \{(i,j) \in I \times J | (d_{ij} \leq \min\{d_{kj} | k \in S_0\} \wedge i \notin S_0) \vee (d_{ij} \leq \min\{\min\{d_{kj} | k \in S_0\}, d_{H(i)j}\} \wedge i \in S_0)\}.$$

It is sufficient to show that, for any $(i,j) \in \overline{L}$ such that $\bar{y}_{ij} = 1$, we also have $(i,j) \in \overline{R}$. For that, let $(i^*, j^*) \in \overline{L}$ with $\bar{y}_{i^*j^*} = 1$, and $k^* = \arg\min_{k \in S_0}\{d_{kj^*}\}$. First, we prove that $d_{i^*j^*} \leq d_{k^*j^*}$. We divide this proof into two cases:

Case 1: $k^* \in S_b$.
For this case, we have $d_{i^*j^*} \leq \min\{d_{kj^*} | k \in S_b\}$ since $(i^*, j^*) \in \overline{L}$, and $\min\{d_{kj^*} | k \in S_b\} \leq d_{k^*j^*}$ since $k^* \in S_b$. Therefore, $d_{i^*j^*} \leq d_{k^*j^*}$.

Case 2: $k^* \notin S_b$.
For this case, $k^* \in S_w$ by the definition of S_b. Hence, $d_{i^*j^*} = \min\{d_{kj^*} | k \in S_w\} \leq d_{k^*j^*}$.

In order to prove that $(i^*, j^*) \in \overline{R}$, it remains to show that $d_{i^*j^*} \leq d_{H(i^*),j^*}$ for the case where $i^* \in S_0$. For this case, $H(i^*) \in S_b$ by the definition of S_b, since $i^* \in S_w$. Hence, $d_{i^*j^*} \leq \min\{d_{kj^*} | k \in S_b\} \leq d_{H(i^*)j^*}$ since $(i^*, j^*) \in \overline{L}$. Therefore, $(i^*, j^*) \in \overline{R}$. □

4 Separation Problem

In this section, we define the separation problem and two separation procedures for the family of constraints given by (5). We also describe a heuristic procedure that, for each generated value of S_0 (not necessarily associated to a violated constraint (5)), gives a corresponding function $H : S_0 \to I$ that maximizes violation of the associated strengthened inequality (8).

The separation problem of (5) is defined as follows. Given a fractional solution $(\overline{z}, \overline{x}, \overline{y}) \in \mathbb{R} \times [0,1]^{|I|} \times [0,1]^{|I| \times |J|}$ that satisfies (2), (3), (4) and some of the constraints (5), the separation problem consists of finding the strategy $S_0 \in S$ that maximizes the violation of (5). For that, S_0 should to minimize the sum

$$\sum_{j \in J} w_j \sum_{i \in I | d_{ij} \leq \min\{d_{kj} | k \in S_0\}} \overline{y}_{ij}. \tag{11}$$

Let us define for each $i \in I$ and $j \in J$, the gain g_{jk} in this way:

$$g_{jk} = w_j \sum_{i \in I | d_{ij} \leq d_{kj}} \overline{y}_{ij} \tag{12}$$

For each customer j and each facility $k \in S_0$, g_{jk} indicates a portion of the sum (11) associated to the customer j if k is the facility of S_0 that is closest to j. Then, the separation problem consists of finding the black's strategy $S_0 \in S$ that minimizes

$$\sum_{j \in J} \min\{g_{jk} | k \in S_0\}. \tag{13}$$

Next, we show an IP formulation for this problem. Let the binary variable s_k indicate that $k \in S_0$. Let also the binary variable t_{jk} indicate that the gain g_{jk} is used for the customer j. The complete formulation is the following:

$$\min \quad g_{jk} \times t_{jk} \tag{14}$$

$$\text{subject to} \quad \sum_{k \in I} s_k = p \tag{15}$$

$$t_{jk} \leq s_k \quad ,\forall j \in J, \forall k \in I. \tag{16}$$

$$\sum_{k \in I} t_{jk} = 1 \quad ,\forall j \in J. \tag{17}$$

$$t_{jk} \in \{0,1\} \quad ,\forall k \in I,\ \forall j \in J. \tag{18}$$

$$s_k \in \{0,1\} \quad ,\forall k \in I. \tag{19}$$

The value of the objetive function (14) is equivalent to the sum (13). Constraint (15) ensures that S_0 has exactly p facilities. Constraints (16) ensure the consistency between the variables t_{jk} and s_k. Finally, constraints (17) ensure that, for each customer j, there is only one facility k such that the gain g_{jk} is considered in (14).

We also propose a greedy heuristic for the separation problem. This heuristic is used to efficiently find some violated cuts avoiding some the IP optimizations. To describe this heuristic, we recall that the separation problem is to find the strategy $S_0 \in S$ which minimizes (13). The heuristic greedily chooses p facilities one at a time as follows. At each iteration, it chooses the facility that causes the minimum increase in the value of (13).

Our exact algorithm is a branch-and-cut, where we apply the separation procedures described before but, instead of adding the constraints (5) to the formulation, we add a corresponding strengthened cut (8) by choosing the function H that maximizes the constraint violation. The choice of H is computed efficiently because the selection of $H(i)$ is done independently for each $i \in S_0$. For each i, we simply choose the value of $H(i)$ that maximizes the sum of all products $w_j \overline{y}_{ij}$ having $d_{H(i)j} < d_{ij}$.

In order to speed up our method, the cuts are separated in two ways: while the gap is greater than 5%, we separate cuts associated to the strengthened constraints 8 for any solution. Otherwise, when the gap is smaller or equal 5%, the cuts are separeted only for integer solutions (for that, we use only the IP based separation). Besides, the branch is preferably performed over the x variables, since the number of such variables is smaller than the number of y variables.

5 Computational Experiments

We tested our method on 10 instances from the benchmark library *Discrete Location Problems*. For all those instances, customers and applicant facilities are in the same sites ($I = J$). Besides, all the distances between applicant facilities and customers are assumed to be euclidean. All the tests are carried out in a 2.13 GHz PC Pentium Intel Core 2 duo with 2 Gb of RAM.

For all the instances considered in this paper, we apply our method using the best value found by the heuristic proposed in Alekseeva *et al.* (2010) as an upper bound. Besides, all the instances have 100 customers. We tested each one of the 10 instances in three different ways: $i)$ $p = 5$ and $w_j = 1$, $ii)$ $p = 5$ and $w_j \in (0, 200)$ and $iii)$ $p = 10$ and $w_j = 1$. The results obtained by our method for the three previous cases are respectively in tables 1, 2 and 3. The following headers are used for the columns: $|J|$ and p indicate the instance characteristics, *Sum of profits* indicates the sum $\sum_{j \in J} w_j$, *Root LB* and *Best UB* indicate respectively the lower bound at the root node and the best known upper bound obtained by Alekseeva *et al.* (2010), *Root gap(%)* indicates the gap between the columns *Root LB* and *Best UB*, *Opt* indicates the black's profit at the optimal solution, *#Sep IP* and *#Sep Greedy* indicate respectively the number of cut separations by IP optimization and the number of greedy cut separations, *#Cuts greedy* and *#Cuts IP* indicate the total number of

Table 1: Summary of our results for $p = 5$ and $w_j = 1$, $j \in J$.

Instance	$\|J\|$	p	Best UB	Root LB	opt	Root gap(%)	#SEP IP	#SEP Greedy	#Cuts Greedy	#Cuts IP	Greedy Time(s)	IP Time(s)	Total Time(s)
111	100	5	53	47	53	11.32	258	329	71	218	514.07	323.83	2144.96
211	100	5	52	46	52	11.54	209	283	74	191	439.14	225.10	1803.56
311	100	5	55	47	55	14.55	864	969	105	728	1519.96	1282.29	22219.41
411	100	5	53	47	53	11.32	328	391	63	274	610.07	477.52	3370.54
511	100	5	53	46	53	13.21	259	314	55	224	488.39	316.87	1710.72
611	100	5	53	47	53	11.32	261	336	75	215	525.22	310.40	2449.33
711	100	5	53	46	53	13.21	319	403	84	281	623.79	374.56	3702.43
811	100	5	52	47	52	9.62	120	196	76	100	307.82	152.62	1304.47
911	100	5	53	47	53	11.32	335	407	72	279	630.98	507.35	2693.97
1011	100	5	53	46	53	13.21	423	518	95	350	809.17	712.75	3904.90

Table 2: Summary of our results for $p = 5$ and $w_j \in (0, 200)$, $j \in J$.

Instance	$\|J\|$	p	Sum of profits	Best UB	Root LB	opt	Root gap(%)	#SEP IP	#SEP Greedy	#Cuts Greedy	#Cuts IP	Greedy Time(s)	IP Time(s)	Total Time(s)
111	100	5	8689	4550	3933	4550	13.56	85	158	73	81	378.22	111.14	1638.45
211	100	5	10520	5698	4597	5698	19.32	105	228	123	103	530.69	127.59	10019.50
311	100	5	9351	5136	4069	5136	20.77	271	472	201	269	1094.48	322.39	34229.00
411	100	5	9927	5249	4478	5249	14.69	111	235	124	109	573.39	135.11	4597.20
511	100	5	10243	5649	4388	5649	22.25	251	445	194	249	1039.47	323.55	30542.70
611	100	5	9518	5035	4279	5035	15.01	87	163	76	85	376.86	100.42	2624.78
711	100	5	11199	6046	4916	6046	18.69	119	263	144	117	607.92	131.92	9048.45
811	100	5	9557	5153	4219	5153	18.13	168	282	114	166	665.59	201.13	9995.02
911	100	5	10396	5696	4669	5696	18.03	233	396	163	231	917.52	266.06	26964.30
1011	100	5	10226	5303	4663	5303	12.07	76	136	60	71	314.91	108.78	1879.41

cuts separated by the greedy algorithm and by solving and IP model, respectively. *Greedy Time*, *IP Time* and *Total Time* indicate the total CPU time in seconds consumed by the greedy cut separation algorithm, the IP optimizations and the complete branch-and-cut algorithm, respectively.

For the case where $p = 5$ and $w_j = 1$, the method optimally solved all the 10 instances in reasonable computational times. The total time consumed was smaller than 4000 seconds for 9 out of 10 instances and the most difficult instance was 311 where the total time consumed was approximately 37 hours. For that instance, the root gap was the largest one (14.55%).

The instances become more difficult when the customer profits are not the same. For this case, note in table 2 that only 4 out of 10 instances were solved in less than 1 hour (3600 seconds). Besides, the total time required to optimally solve the instances 311, 511 and 911 exceeded 20000 seconds.

For the case where $p = 10$ and $w_j = 1$, our method obtained root gaps similar to case where $p = 5$ and $w_j = 1$. However, the total computational time consumed is significantly larger since the number of cuts generated increases. Some instances are hard to solve by our method when p = 10. For example, the time required to solve the instances 311, 511 and 611 exceeded 100000 seconds. On the other hand, we remark that these instances were optimally solved for the first time.

Table 4 shows a comparison between the computational times consumed by our method

Table 3: Summary of our results for $p = 10$ and $w_j = 1$, $j \in J$.

Instance	$\|J\|$	p	Best UB	Root LB	opt	Root gap(%)	#SEP Greedy	#SEP IP	#Cuts Greedy	#Cuts IP	Greedy Time(s)	IP Time(s)	Total Time(s)
111	100	10	50	45	50	10.00	761	412	349	364	1191.41	337.92	8734.40
211	100	10	51	44	51	13.73	1144	738	406	650	1754.36	665.20	15940.29
311	100	10	52	45	52	13.46	1830	1347	483	1176	2808.89	1212.54	106664.80
411	100	10	51	45	51	11.76	1542	1028	514	864	2388.73	981.56	35106.50
511	100	10	52	45	52	13.46	3284	2296	988	1881	5109.11	2075.55	380222.00
611	100	10	53	46	53	13.20	2908	2288	620	1995	4531.13	2344.09	242339.08
711	100	10	51	44	51	13.73	1636	1104	532	974	2546.04	1326.17	43979.20
811	100	10	49	45	49	8.16	775	472	303	418	1213.55	416.05	6105.02
911	100	10	51	45	51	11.76	1999	1506	493	1356	3214.82	1326.17	99207.92
1011	100	10	52	45	52	13.46	1670	1078	592	931	2588.40	1012.12	61839.27

Table 4: Comparison between our method and exact one proposed by Alekseeva *et al*. (2010)

Instance	$\|J\|$	p	Time (s)			
			$w_j = 1, j \in J$		$w_j \in (0, 200), j \in J$	
			Our method	Alekseeva *et al*. (2010)	Our method	Alekseeva *et al*. (2010)
111	100	5	**2144.96**	7200.00	**1638.45**	3900.00
211	100	5	**1803.56**	3600.00	10019.50	**2220.00**
311	100	5	**22219.41**	216000.00	**34229.00**	327600.00
411	100	5	**3370.54**	9000.00	**4597.20**	54000.00
511	100	5	**1710.72**	7200.00	**30542.70**	43200.00
611	100	5	**2449.33**	5400.00	**2624.78**	39600.00
711	100	5	**3702.43**	10800.00	**9048.45**	153000.00
811	100	5	**1304.47**	2520.00	**9995.02**	43200.00
911	100	5	**2693.97**	9600.00	**26964.30**	151200.00
1011	100	5	**3904.90**	9900.00	1879.41	**1800.00**

and by the best exact method found in the literature for each instance with $p = 5$ and the two classes of profit. For each instance, we marked in bold the smallest required time. The runs performed by Alekseeva *et al*. (2010) were carried out in a 1.87 GHz Pentium Intel Core with 2 processors, and 2 GB of RAM.

Note that, even considering the differences in the machine specifications, our method is significantly faster for 17 out of the 20 tested instances. For the instances where and $w_j = 1$, our method was faster for 18 out of 20 instances. The hardest instance for the both methods was 311.

6 Conclusions

In this paper, we proposed an exact method for WSD1VG problem based on an IP formulation with an exponential number of constraints. Moreover, a family of strengthened cuts improved our formulation. These cuts were used in a branch-and-cut algorithm with both heuristic and exact separation procedures.

The reported experiments show that our method was faster than the best exact method found in the literature for almost all instances compared, being able to optimally solve 10 previously open instances.

References

Ahn, H., Cheng, S., Cheong, O., Golin, M. e Van Oostrum, R. (2004), Competitive facility location: the voronoi game. *Theoretical Computer Science*, v. 310, n. 1-3, p. 457–467.

Alekseeva, E., Kochetova, N., Kochetov, Y. e Plyasunov, A. (2010), Heuristic and Exact Methods for the Discrete (r— p)-Centroid Problem. *Evolutionary Computation in Combinatorial Optimization*, p. 11–22.

Cheong, O., Har-Peled, S., Linial, N. e Matousek, J. (2004), The one-round voronoi game. *Discrete and Computational Geometry*, v. 31, n. 1, p. 125–138.

Durr, C. e Thang, N. (2007), Nash equilibria in Voronoi games on graphs. *Algorithms–ESA 2007*, p. 17–28.

Fekete, S. e Meijer, H. (2005), The one-round voronoi game replayed. *Algorithms and Data Structures*, p. 150–161.

Mavronicolas, M., Monien, B., Papadopoulou, V. e Schoppmann, F. (2008), Voronoi games on cycle graphs. *Mathematical Foundations of Computer Science 2008*, p. 503–514.

Noltemeier, H., Spoerhase, J. e Wirth, H. (2007), Multiple voting location and single voting location on trees. *European journal of operational research*, v. 181, n. 2, p. 654–667.

Teramoto, S., Demaine, E. e Uehara, R. Voronoi game on graphs and its complexity. *2006 IEEE Symposium on Computational Intelligence and Games*, p. 265–271, 2006.