UNIVERSIDAD DE LAS AMÉRICAS PUEBLA

ESCUELA DE INGENIERÍAS

DEPARTAMENTO DE INGENIERÍA INDUSTRIAL MECÁNICA Y LOGÍSTICA Y
DEPARTAMENTO DE COMPUTACIÓN ELECTRÓNICA Y MECATRÓNICA

A GRASP-TABU HEURISTIC FOR THE HANDOVER MINIMIZATION PROBLEM

TESIS QUE, PARA COMPLETAR LOS REQUISITOS DEL PROGRAMA DE HONORES PRESENTA EL ESTUDIANTE

CARLOS ARTURO NÚÑEZ FRANCO

150342

DIRECTOR

DOLORES EDWIGES LUNA REYES

SAN ANDRÉS CHOLULA, PUEBLA. PRIMAVERA, 2020

TESIS QUE, PARA COMPLETAR LOS REQUISITOS DEL PROGRAMA DE HONORES PRESENTA EL ESTUDIANTE

CARLOS ARTURO NÚÑEZ FRANCO
150342

DIRECTOR DE TESIS

_____
Dra. Dolores Edwiges Luna Reyes

PRESIDENTE DE TESIS

_____
Dr. Juan Antonio Díaz García

SECRETARIO DE TESIS

_____
Dra. Nelly Monserrat Hernández González

Index

Abstract

In a cellular network a transceiver connects to the base station (tower) from which it receives the strongest signal. If the transceiver is moving it will connect to different base stations across its trajectory. Every base station is controlled by one radio network controller (RNC). A RNC can control multiple base stations, as long as the total traffic associated with the base stations does not exceed the amount of traffic the RNC can handle. One of the RNCs main tasks is transferring an ongoing call or data session from one base station to another in a process called handover. The transfers between base stations belonging to different RNC's take more time and energy and have a higher failure probability than handovers between stations belonging to the same RNC. A failure means the connection gets dropped, thus the Handover Minimization Problem arises in order to find the assignment of the base stations to the RNCs that minimizes the number of handovers between base stations belonging to different RNCs and doesn't violate the traffic capacity constraint of each RNC.

The problem is further described with aid of a brief history of its origin and a model for the problem is presented. In this thesis, we propose a GRASP-Tabu heuristic to find feasible solutions for the problem. A set of small instances randomly generated are used to test the quality of the proposed heuristic. The results obtained using the model and the heuristic are shown and analyzed. The results are of good quality showing that the proposed GRASP-Tabu heuristic works well, however these results can be improved as a future research.

Keywords: Cellular Network, Base Station, RNC, Handover, Heuristic, GRASP, Tabu.

1. Introduction

Networks are everywhere and technological advancements have provided the ever-growing cellular networks the capacity to host more transceivers and reach the most recondite places on earth, bringing millions of people closer with a touch of a finger, allowing for information to travel faster and making it available to more people than ever before.

A cellular network, also known as mobility network, is composed of radio network controllers (RNC), base stations (cell towers) and mobile transceivers such as mobile phones, tablets and computers, among others. The effective area covered by a base station is known as a cell and it's where the tower's signal is stronger than other towers' signals (see Figure 01).

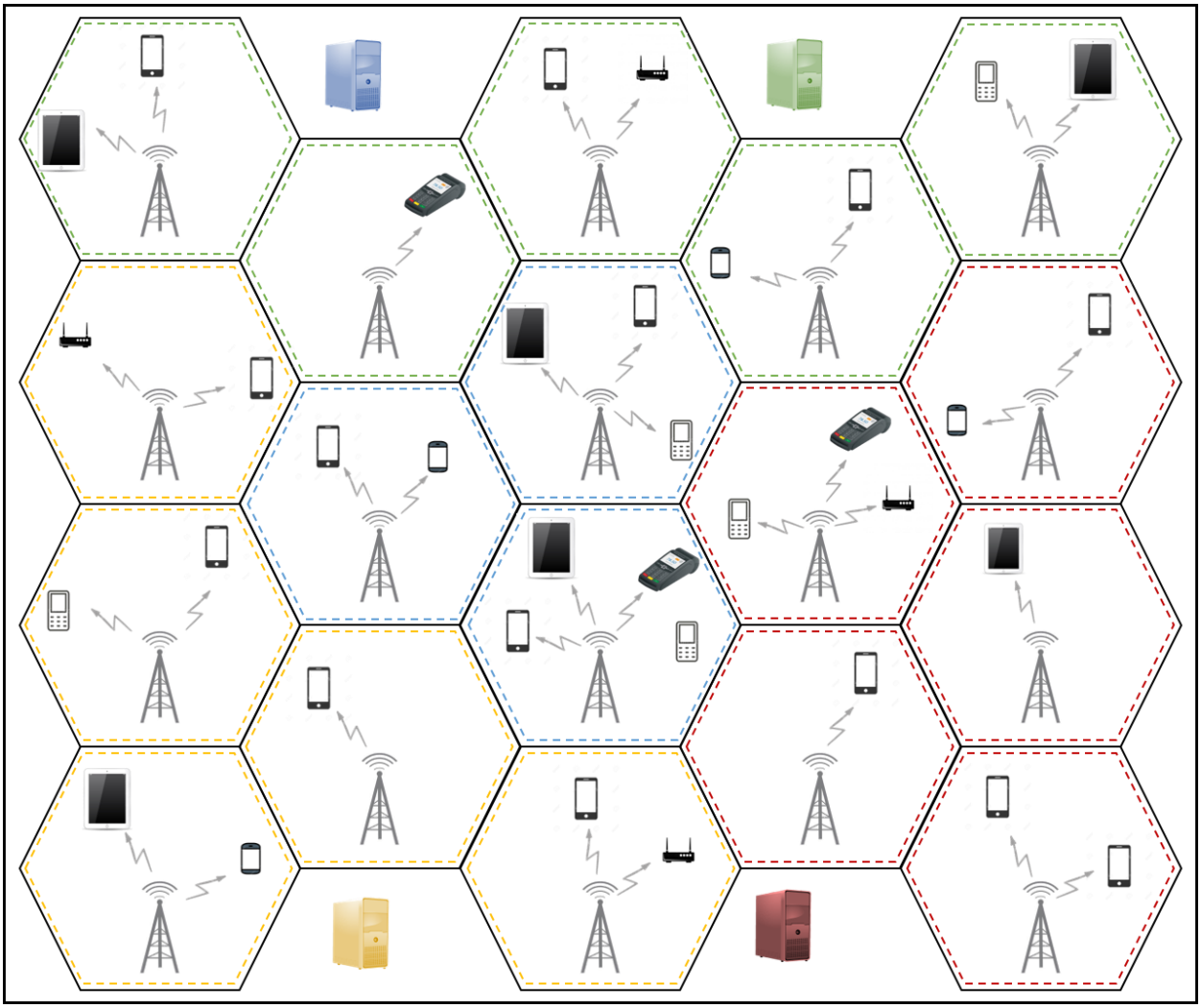


Figure 01. Representation of a cellular network

A mobile transceiver will connect to the one base station it receives the strongest signal, in other words, it will connect to the base station of the cell where it is located. As a device moves through space it may change cells, thus requiring the device to reconnect to the new closest tower. The transfer of connecting from one base stations to another is called handover. (Tekinay and Jabbari, 1991).

Each base station is controlled by the one radio network controller it's connected to. Among the controlled processes are handovers. Handovers between base stations assigned to different RNCs not only require more computing power, therefore more time and energy, than handovers between base stations assigned to the same RNC, but also have a higher fail probability, which means connection gets dropped and directly affects the perceived quality of the service, thus making handovers between different RNCs undesirable and must be kept at a minimum.

An RNC can handle certain amount of traffic that originates from the base stations (RNC's capacity). The amount of traffic on a base station relates directly on how many transceivers are connected to it, meaning a base station in a city center will have more traffic than one on the suburbs. Although the capacities of the RNC are fixed, they may vary between one another and the base station's traffic are not only different between them, but also vary over time.

The handover minimization problem is a real-life problem encountered by engineers in various steps of cellular network design, implementation and operation. The problem consists on assigning all base stations to RNCs in a way that each base station is assigned to only one RNC, the total traffic of base stations assigned to an RNC must equal or lower than the RNC's capacity and the number handovers between base stations connected to different RNCs is minimized. By working with the average handover during daytime the problem becomes deterministic and the obtained solution will perform well on most situations.

Since cellular networks will keep growing both in people using it (active transceivers) and infrastructure to support them (base stations and RNCs) this problem is of particular interest as having the best possible network design is key to reduce operation costs and provide a better customer service. Although there are multiple parameters from which a cost function can be build, the number of handovers between base stations connected to different RNCs is the biggest contributor on both in operation costs and perceived quality of service (Jabbari et al. 1995), thus we used it as function.

This problem is a capacitated clustering problem (CCP) since it consists on forming an specified number of groups (number of RNCs) from a set of elements (base stations) in a way that the sum of the weights (traffic) is within some capacity (RNC capacity) and the sum of the benefit (handovers) between pairs of elements in the same group is maximized. Maximizing the handovers between base stations assigned to the same RNC is the same as minimizing the handovers between base stations assigned to different RNCs, thus it will give the same arrangement.

The following model for solving the CCP was proposed by Moran et al. (2013). Let T be the set of base stations and let $t(i)$ be the total traffic between base station $i \in T$ and the transceivers connected to it. Let R be the set of RNCs and let $c(k)$ be the maximum amount of traffic (capacity) RNC $k \in R$ can handle. Finally, let $h(i, j)$ be the total number of handovers between base stations i and j ($i, j \in T$, $i \neq j$). Note that $h(i, j)$ and $h(j, i)$ may differ.

Consider the graph $G = (T , E)$ where the node-set T represents the base stations and the edge-set E is such that if base stations $i, j \in T$ are such that $h(i, j)+h(j, i) > 0$, then $e = (i, j) \in E$. Let the binary variable $x(e, k) = 1$ if and only if edge e has both of its endpoints assigned to RNC $k \in R$. Furthermore, let the binary variable $y(i, k) = 1$ if and only if base station $i \in T$ is assigned to RNC $k \in R$.

Since minimizing the total handover between base stations assigned to different RNCs is equivalent to maximizing the total handover between base stations assigned to the same RNC, we can redefine the objective function as

$$\max \sum_{k \in R} \sum_{e=(i,j) \in E} h(i,j)x(e,k)$$

Next, regarding the problem's constraints, we first require that each base station $i \in T$ must be assigned to one and only one RNC, i.e.

$$\sum_{k \in R} y(i,k) = 1, \quad \forall\, i \in T$$

The next set of constraints forces $x(e, k) = 0$ if either one of the endpoints of edge e is not assigned to RNC k, i.e.

$$x(e,k) \leq y(i,k), \quad \forall\, e = (i,j) \in E, \quad \forall\, k \in R \tag{1}$$

$$x(e,k) \leq y(j,k), \ \ \forall\, e = (i,j) \in E, \ \ \forall\, k \ \in R \tag{2}$$

$$x(e,k) \geq y(i,k) + y(j,k) - 1, \ \ \forall\, e = (i,j) \in E, \ \ \forall\, k \ \in R \tag{3}$$

Note that constraints (1) and (2) together with the objective function make constraint (3) redundant. That is, when $y(i, k)$ or $y(j, k)$ is 0 then $x(e, k) = 0$ and constraint (3) is satisfied; also, when both $y(i, k)$ and $y(j, k)$ are 1 then $x(e, k) = 1$ since the objective is to maximize the total handover between base stations assigned to the same RNC.

The final set of constraints limits the set of base stations that can be assigned to each RNC. Since RNC k has capacity for at most $c(k)$ units of traffic, the sum of the traffic values of all base stations assigned to RNC k cannot exceed $c(k)$, i.e.

$$\sum_{i \in T} t(i)y(i,k) \leq c(k), \ \ \forall\, k \ \in R$$

After discarding the redundant constraint, the mixed integer program is as follows:

$$\max \sum_{k \in R} \sum_{e=(i,j) \in E} h(i,j)x(e,k)$$

s.t.

$$\sum_{k \in R} y(i,k) = 1, \ \ \forall\, i \ \in \ T$$

$$x(e,k) \leq y(i,k), \ \ \forall\, e = (i,j) \in E, \ \ \forall\, k \ \in R$$

$$x(e,k) \leq y(j,k), \ \ \forall\, e = (i,j) \in E, \ \ \forall\, k \ \in R$$

$$\sum_{i \in T} t(i)y(i,k) \leq c(k), \ \ \forall\, k \ \in R$$

$$x(e,k)$$

The CCP problem is an NP-hard problem (Brucker, 1978), therefore the problem's model will not be able to solve real life instances of the problem. Instead we use the model on small instances in order to get the best possible assignment (optimum) and the time it takes to get it and compare those results with the ones of the proposed heuristic. The goal is that the heuristic solutions are within some low gap percentage, but it runs faster than the model.

In this thesis, we propose a GRASP-Tabu heuristic consisting on loop running a 3-phase greedy for constructing the initial arrangement that may or may not be feasible. Then a Tabu search with two neighborhoods that guaranty a change in the solution structure: base station reassignment and two base station exchange, improves the initial arrangement, also allowing for non-feasible arrangements during the iterations and penalizing said non-feasibility in a way that after some iterations the algorithm returns to a feasible region. After $|R|*|T|*4$ iterations without improvement the solution is returned, and another GRASP iteration begins. The GRASP-Tabu will not stop until 1000 iterations without improvement have passed. The parameters were obtained experimentally.

The objective is that the proposed heuristic is capable of finding feasible solutions such that the average gap for each instance is bellow 5% and that the time is lower than $|R|*|T|/3$ seconds. For this a set of 40 randomly generated instances were run through the model and the heuristic at various steps to keep control of the time and gap.

The remainder of this thesis is organized as follows. Section 2 follows a brief history of the problem and various proposed heuristics to solve it. In Section 3 we describe the proposed GRASP-Tabu heuristic. Experimental results and computational experience are presented in Section 4. And finally, in Section 5 conclusions and further work are presented.

2. Literature Review

The handover minimization problem is an NP-Hard problem that has been studied since the beginning of the 1980's and can be approached as a capacitated clustering problem (CCP).

These problems began to become apparent around 1960's because partitioning a set into mutually exclusive subsets is required in a large variety of areas and applications such as manufacturing, network design, circuit design, pattern recognition, mail delivery, habitat classification, facility location, and statistical data analysis, to name the most prominent (Ahmadi, 2005).

One of the first models proposed for this CCP was proposed by Mulvey and Beck in 1984 for sales force territory design. The problem consisted on finding up to *p*-capacitated clusters such that each cluster has an assigned median related to its customers, and the sum of the difference between each customer assigned to it and the median value is minimized. This model later came to be known as the p-median capacitated clustering problem.

In 2005 Negreiros and Palhano developed the model for CCP called *p*-centered capacitated clustering problem or capacitated (geo) centered clustering problem (CCCP) in which the median is replaced by the centroid. This non-linear model minimizes the sum of dissimilarities in each cluster and allows to either get exactly k clusters or between 2 and k. All clusters having the same capacity. They also developed a heuristic for the CCCP consisting in two parts. The constructive part uses a constructive cluster procedure known as Forgy's algoritm, which builds an initial solution oriented by a log-polynomial algorithm using structured geometric balanced q-trees. Then, the second part uses a variable neighborhood search to improve the initial solution obtained in the first part of the heuristic.

Deng and Bard (2011) proposed a reactive GRASP with path relinking for the same problem. During the first phase, the algorithm initializes the *p* clusters either by using heaviest weight edges algorithm (HWE) or the constrained minimum cut algorithm (CMC). An interesting feature is that the RCL is constructed using both nodes and edges, which according to Festa and Resende (2009), is an unconventional strategy. The second phase explores three types of neighborhood, which are extended node insertion, extended edge insertion and node exchange. Although the GRASP run time increases exponentially with larger instances, the times were still lower than the CPLEX model while also providing very good quality results.

Gallego et al. developed in 2013 a Tabu search with strategic oscillation for the maximally diverse grouping problem (MDGP), which consists on grouping a set of *M* elements into *G* mutually disjoint groups in such a way that the diversity among the elements in each group is maximized. The MDGP is a special case of the CCP in which all the elements have a weight of one unit. Their results proved that their heuristic outperforms the previous approaches for this problem.

Morán et al. (2013) developed three heuristics for the handover minimization problem, which as discussed before, is also a CCP problem. The first one is a GRASP with path-relinking, the second one is a GRASP with evolutionary path-relinking and finally a biased random-key genetic algorithm. The three heuristics had their parameters adjusted using the method described in another of their works (Morán et al., 2013). According to the results, the GRASP with evolutionary path-relinking was the best by getting better results while also doing it faster. On the other hand, the biased random-key genetic algorithm was the slowest and gave the worst results of the three. Although the results weren't so spread from each other the time they took to run was the deciding factor. Their results come from instances generated by an algorithm of their own and had the capacity constrained relaxed by reassigning to each RNC the largest capacity generated, i.e. $c(l) \leftarrow \max\{ c(k) : k \in R \}$, for all $l \in R$.

Martínez, et al. (2015) proposed a GRASP, a Tabu algorithm with a 2-1 exchange neighborhood and a hybrid GRASP-Tabu. They also programmed the algorithms proposed in Deng and Bard's (2011) and in Gallego et al. (2013). They ran two sets of instances, RanReal and DB, with the five heuristics and their results indicated that the performance of the competing methods varies with the data sets. For the set RanReal their Tabu search performed the best with a deviation of 0.15%, in second came their GRASP-Tabu with 0.61%, and third was Gallego's algorithm. For the DB set the algorithm proposed by Gallego et al. (2013) came first with 0.02% of deviation while their Tabu search performed the worst with 4.52%.

Brimberg et al, (2017) developed two algorithms for the CCP problem. The first is a General VNS heuristic (GVNS) and the second one a Skewed VNS (SVNS) approach that permits moves to inferior solutions if they meet specified acceptance criteria with the goal of diversifying the search to other promising regions of the solution space. Both algorithms use a powerful variable neighborhood descent (VND) in the local improvement step with three neighborhoods. The first

one is node relocation, the second 1-1 node exchange and the last a 2-1 node exchange. They tested both algorithms using instances from the set RanReal and the results indicate that the SVNS outperformed the GVNS and even Martínez TS and GRASP-TS.

In 2017 Martínez et al. (2017) revised their previous GRASP and included a new 2-1 exchange neighborhood, then they propose an iterative greedy (IG) and a hybrid IG-GRASP. They also tested a metaheuristic postprocessing method called Math that starts with the solution obtained with the IG-GRASP and then fixes some variables in the integer linear program. This standard method in mathematical programming aims to "refine" a heuristic solution. Their results proved that the hybrid IG-GRASP provided outperformed all previous methods and, that perhaps memory-based construction is an effective mechanism within multi-start heuristic search techniques.

Zhou et al. (2018) presented two heuristics for the CCP. The first one is a Tabu search approach (denoted as FITS) that alternates between exploration in feasible and infeasible search space regions, and a memetic algorithm (MA) that extends FITS with a quality-and-distance pool updating strategy and a dedicated cluster-based crossover. The algorithms were executed with instances from five sets and their results conclude that both compete with current the "state-of-the-art" algorithms discussed previously.

Now we propose a GRASP-Tabu where both the constructive and improvement phases are permitted to leave the feasible region, with some penalization, providing the possibility to reach more solutions from the current iteration. For the Tabu search, 2 neighborhoods seem the most convenient to use, base station reassignment and exchange two base stations.

3. Methodology

The handover minimization problem can be solved using various approaches. In this case we present a heuristic to find feasible solutions for the problem. In this case, we present a GRASP-Tabu heuristic using 3-greedy phases for generating the starting solutions. Then to improve the initial solutions, we use a Tabu search that explores two neighborhoods: base station reassignment and two base station exchange.

A heuristic is an algorithmic method that does not guarantee to obtain the optimal solution but is able to produce a high-quality solution within a time frame when exact methods (such as branch and cut, branch and bound, etc.) are too slow, inefficient, or not able to find the optimal solution. A high-quality solution will be a feasible scenario represented by the combination of variables that satisfy all constraints and has low gap with respect to the optimal solution. The disadvantages of using a heuristic when there are several feasible solutions comes from the combination the probability distribution for sampling all feasible solutions given the used parameters, if the optimum is reached or has high reach probability and if it's really needed, the size of the confidence interval of the solutions and the execution time.

A greedy randomized adaptive search procedure (GRASP) is a metaheuristic algorithm proposed by Resende and Feo in 1989 and formalized in 1995. It is based on an iterative process, which starts by building a randomized greedy starting solution which then goes through iterative improvements from a local search phase.

A Tabu search is another metaheuristic approach that is more powerful than a local search because it avoids being stuck in a local-optimum by allowing moving to a worst solution when no other better solution was found in the current neighborhood. The algorithm uses a list of prohibited movements (tabu list) in order to avoid returning to already visited solution. It is also possible to allow movements that get solutions out of the feasible region by adding penalization functions that will return the solutions to the feasible region after a few moves.

Let's have in mind the notation used by the model presented in the introduction, the notation in the instance generator algorithm discussed in the next chapter and remembering that the handover minimization problem can be represented as graph $G=(T,E)$ where T is the set of base stations (towers) and E the set of edges, such that if $i, j \in T$ and $h(i, j) + h(j, i) > 0$, then $e = (i, j) \in E$.:

The GRASP-Tabu works the following way. First $|T|, |R|$, $t(i)$, $h(i,j)$ and $c(k)$ are read from a .txt file. Then, the randomized greedy function is called, the input parameter is α and an initial assignment set is returned. Later, the TabuSearch function is called with the tabu tenure (tt), the penalization factor (pf) and the initial assignment set ($x0$) as input arguments, and an assignment set and its handover function value (h) are returned, this solution is now the best solution. Now the iterative process begins where the stop criterion is that there hasn't been improvement in the last 1000 iterations. During this process we get another initial arrangement set, then the solution is improved and returned with its handover cost function and that value is compared with the lowest one found, if the value is lower, it means the assignment is better, we replace the previous best with the new solution and reset the criterion count (see Algorithm 01).

Notice how the parameters alpha (α), tabu tenure (tt) and penalization factor (pf) change in a loop taking a wide combination of values and providing a broader solution scope. The alpha value was chosen to move between 0.5 and 0.95 with a step of 0.05, because for most instances of size 20x5, 30x5, 40x5 and 40x10 the best initial solutions were obtained with an alpha value above 0.85, but for sizes 20x10, 30x15 and 40x20 a more randomized approach (with values between 0.5 and 0.7) performed the best. In the case of the tabu tenure, we notice how the more RNCs there are the more probability the algorithm must be trapped in a cycle. Testing the algorithm with different tabu tenure values, we did not find a value that would work for all instances. Therefore, again the best strategy is to vary the parameter value between $|R|$ and $2 * |R|$, the more RNCs there are, the higher the value and the value changes for every iteration thus we avoid getting stuck in cycles. Finally, for the value of the penalization factor parameter, we first observed how the more towers there are the higher the handover would be. The first though was making pf a large number that forces the algorithm to return to feasibility as soon as possible, but that didn't work since once feasibility was reach every unfeasible assignment had a bh value above the current one. By also changing this parameter between a low value (T) that allows leaving the feasible region and a higher value ($2 * |T|$) that allows entering feasibility quickly the best results were produced. However, the parameter values need refinement in a way that the values depend more on size and have less combinations but more powerful ones. Perhaps a design of experiments or a method like the one described in Morán 2013.

Algorithm 01. GRASP-Tabu

```
1   ReadData(instance)
2   α ← 0.5
3   tt ← |R|
4   pf ← |T|
5   x0 ← RanGreedy(α)
6   y, h ← TabuSearch(tt, pf, x0)
7   count ← 0
8   while (count < 1000) do
9       if (α < 0.95) then
10          α ← α + 0.05
11      else
12          α ← 0.5
13          if (tt < 2 * |R|) then
14              tt ← tt + 1
15          else
16              tt ← |R|
17              if (pf < 2 * |T|) then
18                  pf ← pf + 1
19              else
20                  pf ← |T|
21              end − if
22          end − if
23      end − if
24      x0 ← RanGreedy(α)
25      y2, h2 ← TabuSearch(tt, pf, x0)
26      if (h2 < h) then
27          y ← y2
28          h ← h2
29          count ← 0
30      end − if
31  end − do
32  DisplayResults(y)
```

Now, we explain the randomized greedy algorithm with more detail. The randomized greedy algorithm is formed by three stages or phases. The first phase initializes each RNC with a pair of base stations. The second phase assigns the remaining base stations, if possible, to RNCs.

The third phase is only necessary if after the second phase there are still unassigned base stations. Phase 2 fails when it is not possible to assign all base stations due to the limited capacity of the RNCs. Therefore, in phase 3, the unassigned base stations are assigned to a RNC regardless their capacity limit.

L V be the set of unassigned base stations, EV be the set of available edges for insertion. In the first phase of the greedy algorithm $\forall\, k \in R$ the first greedy function for the available edge is evaluated and the RCL is build where $g1_{max} = max\{g1(e) \mid e \in EV\}$ and $\alpha \in R$ such that $0 \leq \alpha \leq 1$.

$$\forall\, e \in EV \quad g1(e) \leftarrow h(i,j) + h(j,i)$$

$$RCL_1 = \{e \in EV \mid g1(e) \geq g1_{max} * \alpha\}$$

From the RCL_1 an edge is selected at random, both ends (base stations) are assigned to RNC k and removed them from set V. The set EV is updated such that if $i,j \in V$ and $h(i,j) + h(j,i) > 0$, then $e = (i,j) \in EV$ and the RNC's current capacity is now: $cc(k) \leftarrow t(i^*) + t(j^*)$, where $cc(k)$ is the current capacity of RNC $k, \forall\, k \in R$ and $t(i)$ is the traffic of base station i (see Figure 02). Notice that if $\alpha = 0$ the selection will be completely at random and with $\alpha = 1$ the selection will be greedy.

The second greedy phase starts with every RNC assigned with two base stations. Once again $\forall\, k \in R$ while there is space in current RNC k the second greedy function for the available base stations is evaluated and the RCL_2 is constructed where $g2_{max} = max\{g2(i) \mid i \in V \; and \; cc(k) + t(i) \leq c(k)\}, \forall\, i \in T, RK(k)$ is the set of base stations assigned to RNC k and $c(k)$ is the maximum capacity of RNC $k, \forall\, k \in R$.

$$\forall\, i \in V \quad g2(i) \leftarrow \sum_{j \in RK(k)} (h(i,j) + h(j,i))$$

$$RCL = \{i \in V \mid g2(i) \geq g2_{max} * \alpha \; and \; cc(k) + t(i) \leq c(k)\}$$

Then, from RCL_2 a base station is randomly selected, assigned to RNC k, removed from set V and the RNC's current capacity is now: $cc(k) \leftarrow cc(k) + t(i^*)$. If the RCL_2 list is empty either all base stations have been assigned in previous steps or the current RNC has no room left

for another base station. Therefore, we change k to the next RNC, or if all have been inspected, we end phase 2.

Notice that during this phase the restricted candidate list only includes base stations if assigned to current RLC k the current capacity would be equal or lower than the limit: $cc(k) \leq c(k)$. This can lead to leaving base stations unassigned and not being able to get a feasible solution, thus another phase is required.

The third phase of the randomized greedy algorithm starts while at least one base station has not been assigned to an RNC. The greedy function evaluates the handover sum between a member of set V and the base stations assigned to a member of set R. For the RCL_3 we have ordered pairs (*i*,*k*) where $g3_{max} = max\{g3(i,k) \mid \forall\, i \in V, k \in R\}$.

$$\forall\, i \in V, k \in R \quad g3(i,k) \leftarrow \sum_{j \in RK(k)} (h(i,j) + h(j,i))$$

$$RCL_3 = \{\forall\, i \in V, k \in R \mid g3(i,k) \geq g3_{max} * \alpha\}$$

Then, from the RCL_3 a base station is randomly selected, assigned to RNC k, removed from set V and the RNC's current capacity is now: $cc(k) \leftarrow cc(k) + t(i^*)$ (see Algorithm 02).

Algorithm 02. RanGreedy

```
1   Function RanGreedy(α)
2   V ← T
3   EV ← e = (i,j) | i,j ∈ T and h(i,j) + h(j,i) > 0
4   ∀ k ∈ R do
5       ∀ e ∈ EV   g1(e) ← h(i,j) + h(j,i)
6       g1_max = max{g1(e) | e ∈ EV}
7       RCL = {e ∈ EV | g1(e) ≥ g1_max * α}
8       i*,j* ← RandomSelect(RCL)
9       RK(k) ← RK(k) ∪ {i*,j*}
10      V ← V\{i*,j*}
11      EV ← e = (i,j) | i,j ∈ V and h(i,j) + h(j,i) > 0
12      cc(k) ← t(i*) + t(j*)
13      k ← k + 1
14  end − do
15  ∀ k ∈ R do
16      ∀ i ∈ V   g2_i ← Σ_{j ∈ RK_k}(h_ij + h_ji)
```

17 $g2_{max} = max\{g2(i) \mid i \in V \text{ and } cc(k) + t(i) \leq c(k)\}$
18 $RCL = \{i \in V \mid g2(i) \geq g2_{max} * \alpha \text{ and } cc(k) + t(i) \leq c(k)\}$
19 $if\ (|RCL| > 0)\ then$
20 $i^* \leftarrow RandomSelect(RCL)$
21 $RK(k) \leftarrow RK(k) \cup \{i^*\}$
22 $V \leftarrow V \backslash \{i^*\}$
23 $cc(k) \leftarrow cc(k) + t(i^*)$
24 $else$
25 $k \leftarrow k + 1$
26 $end - if$
27 $end - do$
28 $while\ (V \neq \{\})\ do$
29 $\forall\ i \in V, k \in R \quad g3(i,k) \leftarrow \sum_{j \in RK(k)} (h(i,j) + h(j,i))$
30 $g3_{max} = max\{g3(i,k) \mid \forall\ i \in V, k \in R\}$
31 $RCL = \{\forall\ i \in V, k \in R \mid g3(i,k) \geq g3_{max} * \alpha\}$
32 $i^*, k^* \leftarrow RandomSelect(RCL)$
33 $RK(k^*) \leftarrow RK(k^*) \cup \{i^*\}$
34 $V \leftarrow V \backslash \{i^*\}$
35 $cc(k^*) \leftarrow t(i^*)$
36 $end - do$
37 $return\ RK$

In order to better explain the randomized greedy algorithm, we present an example. Suppose we have the following graph with 25 base stations. Some pairs of base stations will have handovers between them, represented as the 46 edges (see Figure 02).

Pre-RCL

e	g1(e)
(1, 6)	20
(15, 16)	15
(11, 13)	15
(22, 23)	14
(13, 14)	13
(3, 8)	13
(22, 24)	9
(18,25)	9

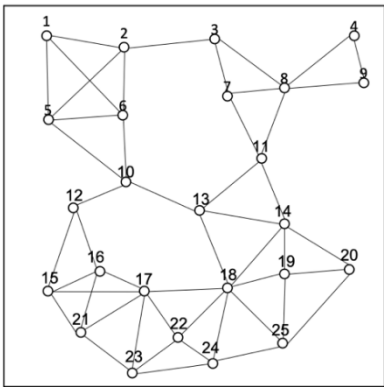


Figure 02. Example Graph

Suppose there are 5 RNCs with limited capacity in which to accommodate the base stations and the alpha is 0.75. The first greedy phase begins by making $k = 1$, meaning we are assigning base stations to the first RNC, then from the available edges for insertion we get the edge with

most handovers, multiply said value by 0.75 and construct the RCL_1 with the available edges for insertion with handovers greater or equal than the obtained value. Now from the RCL we randomly select one edge, assign both ends (base stations) to the first RNC, update the first RNCs current capacity, update the available edges for insertion and increase k by one (see Figure 03).

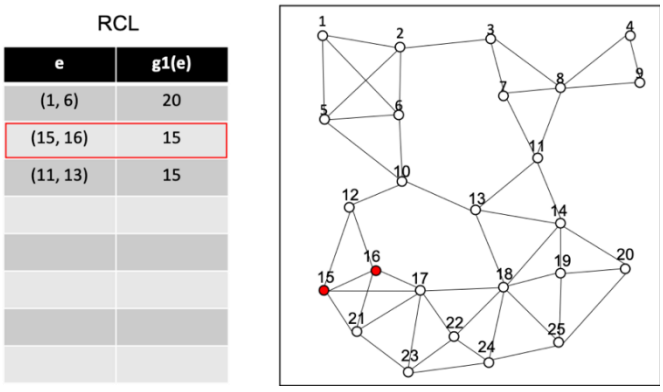


Figure 03. First phase randomized greedy algorithm with $k = 1$.

Now in an iterative process, if k is equal or lower than the number of RNCs ($|R|$) once again from the available edges for insertion we get the one with the highest numbers of handovers, multiply the value by 0.75 and construct the RCL_1 with the available edges with a number of handovers greater or equal than the found value. From the RCL_1 we randomly select one edge, assign both ends (base stations) to the current RNC, update the current RNCs current capacity, update the available edges for insertion and increase k by one (see Figure 04).

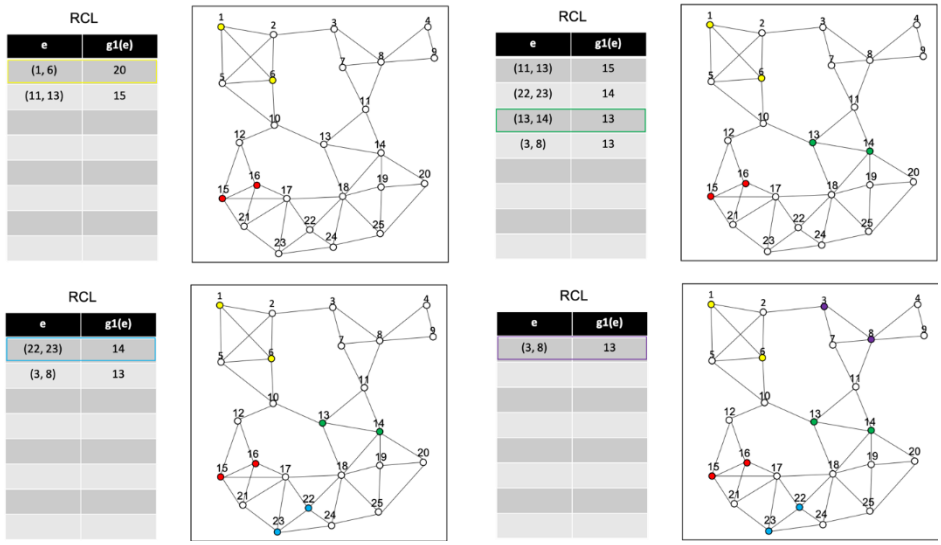


Figure 04. First phase randomized greedy algorithm with $k = 2$ to $k = 5$.

Now, the second phase of the randomized greedy algorithm begins by assigning $k = 1$. The second greedy function is computed for each unassigned base station that can be assigned to the first RNC, in other words, if base station i hasn't been assigned yet and the current capacity of the first RNC plus the traffic associated with base station i is equal or lower than the limit capacity of the RNC, then the greedy function is computed for base station $i, \forall\, i \in V$. Now from the computed values, we get the maximum, multiply it by 0.75 and construct the RCL_2 with every base station in V that has got a $g2$ value equal or higher than the calculated one. We randomly select a base station from the RCL_2, assign it to the first RNC, update the current capacity for the RNC, update V and repeat the process. If there are no candidates in the restricted candidate list it means none of the unassigned base stations "fit" in the RNC, so we increase k by 1 (see Figure 05).

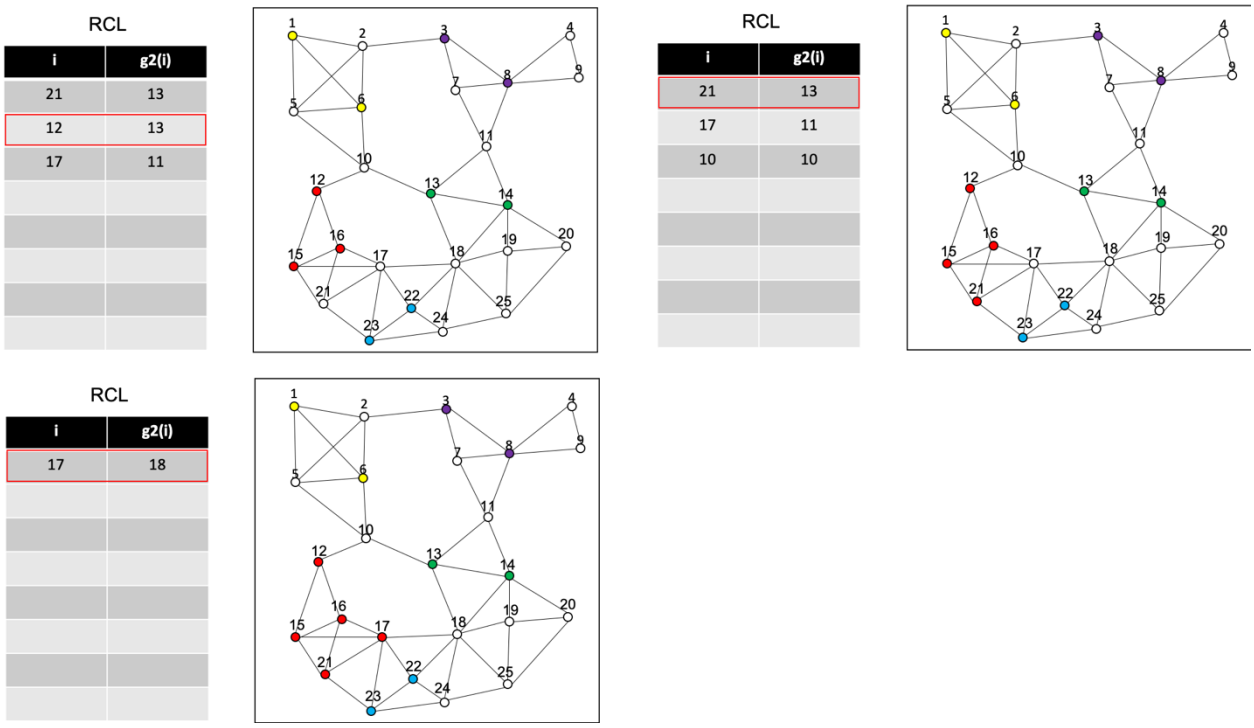
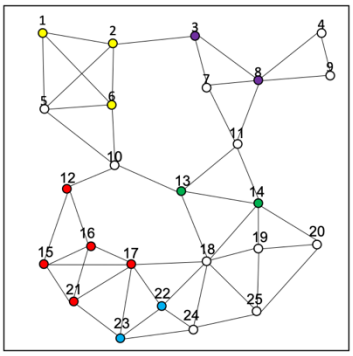


Figure 05. Second phase randomized greedy algorithm with $k = 1$.

The iterations continue while k is equal or lower than the number of RNCs. The second greedy function is computed for each unassigned base station that can be assigned to the current RNC and the RCL_2 is built the same way as before. If there is at least one base station, we randomly select one, assign it to the current RNC, update the current capacity for the RNC, update V and repeat the process. If there are no candidates in RCL_2 k is increased by 1 (see Figures 06 – 09).

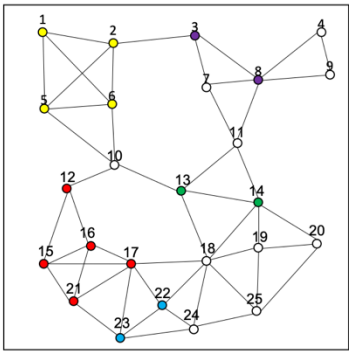
RCL

i	g2(i)
2	13
5	12



RCL

i	g2(i)
5	12



RCL

i	g2(i)
10	13

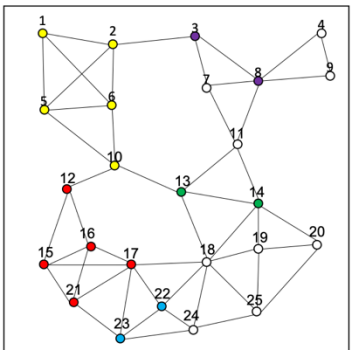
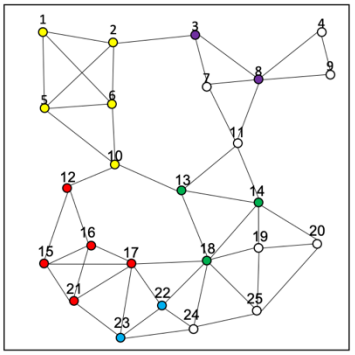


Figure 06. Second phase randomized greedy algorithm $k = 2$.

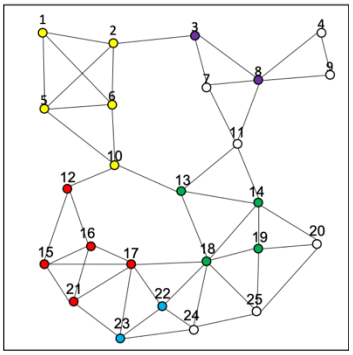
RCL

i	g2(i)
11	11
18	10



RCL

i	g2(i)
19	12
11	11



RCL

i	g2(i)
4	0

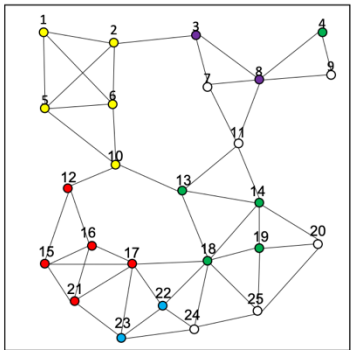
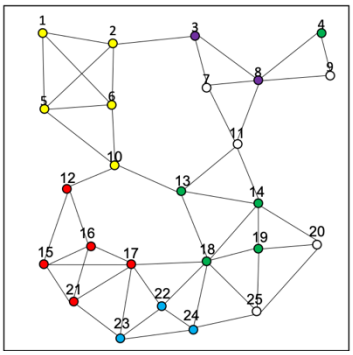


Figure 07. Second phase randomized greedy algorithm with $k = 3$.

RCL

i	g2(i)
24	10



RCL

i	g2(i)
25	6

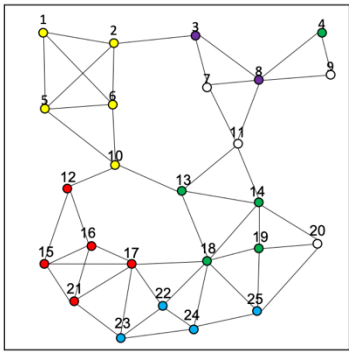
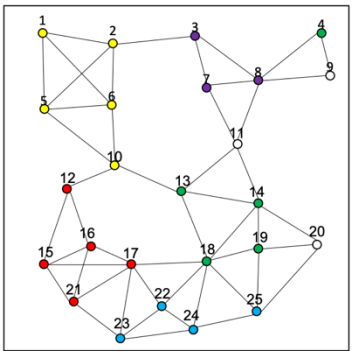


Figure 08. Second phase randomized greedy algorithm with $k = 4$.

RCL

i	g2(i)
7	13



RCL

i	g2(i)
11	11

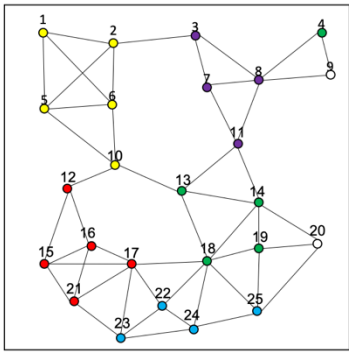


Figure 09. Second phase randomized greedy algorithm with $k = 5$.

The last phase of the randomized greedy algorithm begins if there are still base stations in V. While V is not empty, we compute the third greedy function for each unassigned base station. This time, the RCL_3 is constructed by pairs of (base station – RNC) of the values equal or higher than 0.75 times the maximum value in the matrix $g3(i,k), \forall\, i \in V, \forall\, k \in R$ (see Figures 10-11).

Pre-RCL

i, k	g3(i, k)
20, 3	13
9, 5	7
20, 4	6

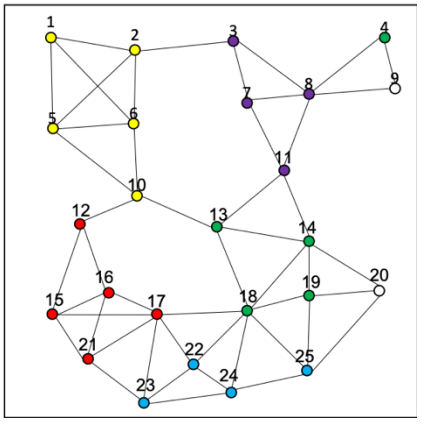
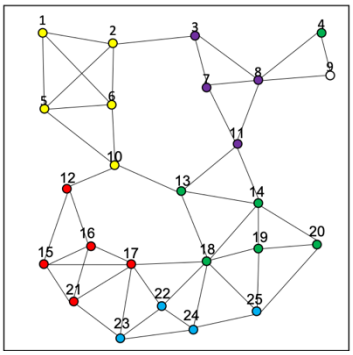


Figure 10. Third phase randomized greedy algorithm: List of possible candidates.

RCL

i, k	g3(i, k)
20, 3	13



RCL

i, k	g3(i, k)
9, 5	7

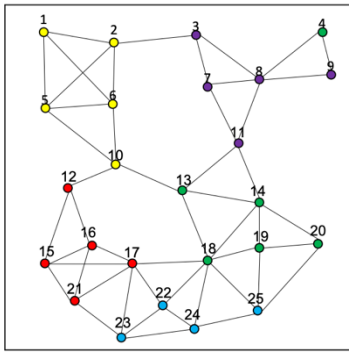


Figure 11. Third phase randomized greedy algorithm: RCL_3.

The randomized greedy function ends after all base stations have been assigned to an RNC. Notice how this process can give non feasible assignments since it allows the violation of the capacity constraint in phase 3.

The initial assignment *x0 is* improved using a Tabu search that explores two neighborhoods and allows unfeasible solutions. In the example the Tabu should find the optimum arrangement (see Figure 12). The first neighborhood (N1) consists on exploring the possibility of moving a base station assigned to one RNC to a different RNC. The improvement will be measured by the difference between the current bh and a new bh which is the value of handover function plus any penalizations from exceeding at least one of the RNCs capacities. The second neighborhood (N2) consists on testing the benefit derived from all possible interchanges between base stations assigned to different RNCs. Once again, an improvement will be the difference between the current bh and a new bh.

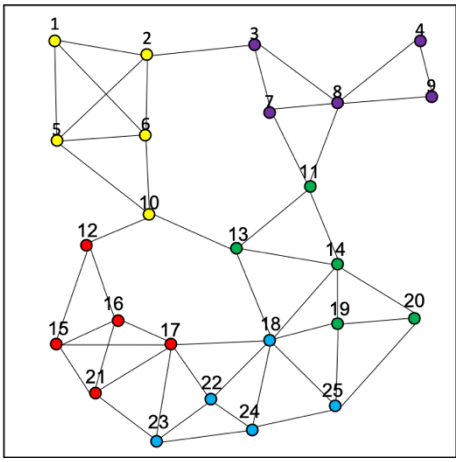


Figure 12. Optimum assignment after the Tabu function.

Both neighborhoods will give the best assignment found, from N1 $x1$ and from N2 $x2$. Then $bh(x1)$ represents bh for the assignment $x1$ and $bh(x2)$ represents bh for the assignment $x2$, so the assignment with lower bh will replace $x0$. If the new bh is lower than the previous one, the criterion parameter is reset by making $iterf = iter$.

If after the replacement $pen(x0) = 0$ then the new assignment is feasible, so the solution is saved. After $|R| * |T| * 4$ iterations without improvement the best solution where the penalization was 0 is returned, and the results are displayed. Remember that the assignments are stored on an array of sets $RK(k), \forall\, k \in R$ (see Algorithm 03).

Algorithm 03. TabuSearch

```
1   Function TabuSearch(tt, pf, x0)
2   iter ← 0
3   Set tabu list.
4   while (iter − iterf < |R| * |T| * 4) do
5       x1 ← Search(N1, pf)
6       x2 ← Search(N2, pf)
7       if (bh(x1) < bh(x2))then
8           x ← x1
9       else
10          x ← x2
11      end − if
12      if (bh(x) < bh(x0)) then
13          iterf ← iter
14      end − if
15      x0 ← x
16      Update tabu list.
17      if (pen(x0) = 0) then
18          y = x0
19      end − if
20  end − do
21  return y, bh
```

4. Results and Discussion

In order to test the efficiency, robustness and quality of the solutions obtained by the proposed GRASP-Tabu heuristic for the handover minimization problem a set of instances is needed, thus we used the instance generator algorithm described in Moran et al (2013), which is able to synthetically generate instances for this problem that mimic conditions encountered in practice.

The instance generator algorithm works by receiving a number of base stations [$|T|$] and a number of RNCs [$|R|$], then it randomly locates the base stations in a unit square and randomly assigns traffic to each one. Next, the algorithm assigns the handover between two base stations as a function that is inversely proportional to the distance between the base stations, unless the distance is greater than a given parameter [r]. The RNC capacities are assigned having in mind that they must accommodate all base stations, thus the sum of the RNCs capacities must be greater than the sum of the base stations traffic.

The parameters used in the generation process are $r = 0.17$ (maximum handover distance), $u_t = 50$ and $l_t = 5$ (upper and lower bound on base station traffic), $u_h = 200$ and $l_h = 5$ (upper and lower bounds on number of handovers), and $u_c = 1.15$ and $l_c = 1.05$ (upper and lower bounds on capacity) such that $u_c > l_c > 1$. The steps of the generation process are:

1. Generate uniformly at random in the unit square the x and y coordinates for each base station $i \in T$.
2. For each base station $i \in T$, generate its traffic uniformly at random between its lower and upper bounds, i.e. $t(i) = randunif(l_t, u_t)$.
3. Let $d(i,j)$ be the distance between stations $i,j \in T$. For each pair of base stations $i,j \in T$ such that if $d(i,j) \leq r$, generate an amount of handover $h(i,j)$ equal to $(l_h - u_h)/r^2 * d^2(i,j) + u_h$. For all $i,j \in T$ such that $d(i,j) > r$, we have that the handover $h(i,j) = 0$.
4. For each RNC $j \in R$, generate a capacity $c(j)$ equal to $randunif(l_c, u_c) \times \bar{t}$, where $\bar{t} = \sum_{i \in T} t(i) / |T|$.

Note that this method does not guarantee that every instance generated has a feasible solution since it is possible that either one base station traffic is greater than all of the RNC's capacities or that there is no possible assignment of base stations to the RNC's without violating at least one capacity. Morán suggested that in order to relax the capacity constraint we reassign to each RNC

the largest capacity generated, i.e. $c(j) \leftarrow max\{c(k) : k \in R\}$, for all $j \in R$. However, since real life problems do not always have the same capacity on every RNC, we did not change the capacities, instead we tested every instance generated so that we work only with instances that have a solution, thus making the problem harder to solve.

We generated 5 instances of each of the following combinations (|*T*|, |*R*|): (20, 5), (20, 10), (30, 5), (30, 10), (30, 15), (40, 5), (40, 10) and (40, 15). Every instance was run on the model and if it wasn't feasible, we generated another one, test it and replace it if feasible. This assured that each of the 40 instances could be solved.

The model was programed on FICO-Xpress and the GRASP-Tabu in C. Both the model and the heuristic ran on a MacBook Pro 2015 running macOS Catalina 10.15 with a 2.9 GHz Dual-Core Intel Core i5 processor and 8GB, 1867 MHz, DDR3 RAM.

The results from running the 40 instances using the model with a time limit of 12 hours are shown on Table 01 The first column refers to the name of the data file, the size column refers to the number of base stations and RNCs ($|T|x|R|$)- The handover column is the objective function value for the optimal or best arrangement found and is the average number of handovers between base stations connected to different RNCs. Finally, the time column represents the elapsed seconds from beginning to end. Note that handover values highlighted are thought to be the optimum, but the model failed narrow the gap within a given time.

Table 1. Xpress run on small instances

Name	Size	Handover	Time (s)	Name	Size	Handover	Time (s)
1517869674	20x5	0.00	0.033	1570986545	30x15	4443.90	3470.481
1517883382	20x5	29.71	0.087	1528117980	30x15	3221.42	43200.000
1528117634	20x5	280.96	0.169	1528117982	30x15	4879.61	0.902
1528117639	20x5	304.16	0.224	1528117984	30x15	4915.94	43200.000
1528117642	20x5	0.00	0.101	1528117986	30x15	3982.40	407.566
1566524515	20x10	2411.93	0.280	1528118414	40x5	806.87	1.552
1528117718	20x10	1376.77	0.245	1528118418	40x5	176.34	1.117
1570986386	20x10	1943.77	3.747	1528118421	40x5	134.84	0.586
1566524519	20x10	1224.46	0.538	1528118423	40x5	383.89	0.872
1570986392	20x10	3729.97	13.589	1528118426	40x5	323.74	1.215
1528117794	30x5	142.33	0.433	1528118631	40x10	3705.70	43200.000
1528117796	30x5	313.88	0.452	1528118633	40x10	3490.48	4874.169
1528117797	30x5	115.36	0.292	1528118635	40x10	2943.00	436.970
1528117798	30x5	1155.85	0.955	1528118637	40x10	1658.47	13.305
1528117799	30x5	592.40	0.547	1528118638	40x10	4519.55	997.222
1528117858	30x10	1025.69	2.488	1528118658	40x15	5937.38	43200.000
1528117859	30x10	1106.54	12147.939	1528118660	40x15	6076.56	43200.000
1528117861	30x10	2245.92	54.619	1528118662	40x15	8712.96	52.000
1528117862	30x10	2392.82	8.181	1528118663	40x15	5157.67	107.000
1528117864	30x10	1788.50	504.368	1528118665	40x15	5162.92	43200.000

* Highlighted results indicate Xpress failed to prove optimality.

On Table 02, the handover values from running the heuristic 10 times for each instance are presented and compared with the best-known value. As we can observe, the proposed heuristic was able to find the optimal or best-known solution in at least one of the 10 run for 38 out of the 40 instances. We can observe that for most instances the heuristic behaved with low variability. The worst case was for instance 1528118421 where the minimum gap was 0%, meaning the optimum was achieved, and the maximum gap was 54%, also the highest gap. However, this high gap percentage van be attributed to the low handover value for the optimal arrangement.

The table seems to indicate that for instances where there are more than 2 base stations per RNC ($|T|/|R| > 2$) the gap is lower than those where the opposite happens ($|T|/|R| \leq 2$). This works at our advantage since real life instances have a greater number of base stations per RNC.

Table 02. Heuristic run on small instances: Handovers

Instance		Xpress	Heuristic					
Name	Size	Handovers	Handovers (min)	Handovers (avg)	Handovers (max)	Gap (min)	Gap (avg)	Gap (max)
1517869674	20x5	0.0	0.0	0.0	0.0	0%	0%	0%
1517883382	20x5	29.7	29.7	29.7	29.7	0%	0%	0%
1528117634	20x5	281.0	281.0	281.0	281.0	0%	0%	0%
1528117639	20x5	304.2	304.2	304.2	304.2	0%	0%	0%
1528117642	20x5	0.0	0.0	0.0	0.0	0%	0%	0%
1566524515	20x10	2411.9	2411.9	2411.9	2411.9	0%	0%	0%
1528117718	20x10	1376.8	1376.8	1376.8	1376.8	0%	0%	0%
1570986386	20x10	1943.8	1943.8	2031.8	2053.7	0%	4%	5%
1566524519	20x10	1224.5	1224.5	1224.5	1224.5	0%	0%	0%
1570986392	20x10	3730.0	3730.0	3869.1	4049.4	0%	4%	8%
1528117794	30x5	142.3	142.3	142.3	142.3	0%	0%	0%
1528117796	30x5	313.9	313.9	313.9	313.9	0%	0%	0%
1528117797	30x5	115.4	115.4	115.4	115.4	0%	0%	0%
1528117798	30x5	1155.9	1155.9	1155.9	1155.9	0%	0%	0%
1528117799	30x5	592.4	592.4	592.4	592.4	0%	0%	0%
1528117858	30x10	1025.7	1025.7	1025.7	1025.7	0%	0%	0%
1528117859	30x10	1106.5	1106.5	1106.5	1106.5	0%	0%	0%
1528117861	30x10	2245.9	2245.9	2307.8	2340.9	0%	3%	4%
1528117862	30x10	2392.8	2392.8	2429.2	2469.0	0%	1%	3%
1528117864	30x10	1788.5	1788.5	1788.5	1788.5	0%	0%	0%
1570986545	30x15	4443.9	4443.9	4443.9	4443.9	0%	0%	0%
1528117980	30x15	3221.4	3221.4	3221.4	3221.4	0%	0%	0%
1528117982	30x15	4879.6	4964.3	5102.2	5421.4	2%	4%	10%
1528117984	30x15	4915.9	4915.9	4915.9	4915.9	0%	0%	0%
1528117986	30x15	3982.4	3982.4	3990.9	4067.6	0%	0%	2%

1528118414	40x5	806.9	806.9	806.9	806.9	0%	0%	0%
1528118418	40x5	176.3	176.3	179.2	205.1	0%	2%	14%
1528118421	40x5	134.8	134.8	202.8	284.2	0%	34%	53%
1528118423	40x5	383.9	383.9	383.9	383.9	0%	0%	0%
1528118426	40x5	323.7	323.7	327.0	356.6	0%	1%	9%
1528118631	40x10	3705.7	3705.7	3725.4	3772.5	0%	1%	2%
1528118633	40x10	3490.5	3490.5	3509.8	3617.9	0%	1%	4%
1528118635	40x10	2943.0	2943.0	2943.0	2943.0	0%	0%	0%
1528118637	40x10	1658.5	1658.5	1658.5	1658.5	0%	0%	0%
1528118638	40x10	4519.6	4519.6	4519.6	4519.6	0%	0%	0%
1528118658	40x15	5937.4	6320.5	6600.8	6987.3	6%	10%	15%
1528118660	40x15	6076.6	6076.6	6076.6	6076.6	0%	0%	0%
1528118662	40x15	8713.0	8713.0	8713.0	8713.0	0%	0%	0%
1528118663	40x15	5157.7	5157.7	5160.0	5169.5	0%	0%	0%
1528118665	40x15	5162.9	5174.0	5181.7	5226.8	0%	0%	1%

On Table 03, we show and compare the time it took to run the heuristic vs the time it took to run the model. Notice how for the model some instances can take a lot of time to solve while others take just a fraction of a second. This variability is addressed on the heuristic where the time it takes to run depends mostly of the size thus time variability between instances of the same size is kept low. By having a predictable time one can be sure at least one good feasible solution will be obtained with the heuristic when needed.

Table 03. Heuristic run on small instances: Times

Instance		Xpress	Heuristic		
Name	Size	Time (s)	Time (s) (min)	Time (s) (avg)	Time (s) (max)
1517869674	20x5	0.0	4.3	4.5	4.6
1517883382	20x5	0.1	4.3	5.0	6.9
1528117634	20x5	0.2	4.3	5.7	7.9
1528117639	20x5	0.2	4.3	5.0	6.3
1528117642	20x5	0.1	4.3	4.4	4.9
1566524515	20x10	0.3	11.1	13.3	16.5

1528117718	20x10	0.2	11.3	11.7	12.3
1570986386	20x10	3.7	11.7	13.1	16.3
1566524519	20x10	0.5	11.5	13.6	17.5
1570986392	20x10	13.6	11.9	15.1	24.2
1528117794	30x5	0.4	10.9	12.1	16.0
1528117796	30x5	0.5	10.9	12.4	16.0
1528117797	30x5	0.3	11.1	13.4	16.2
1528117798	30x5	1.0	9.7	11.8	13.4
1528117799	30x5	0.5	11.1	13.8	20.5
1528117858	30x10	2.5	27.6	33.4	44.3
1528117859	30x10	12147.9	28.5	33.6	38.6
1528117861	30x10	54.6	27.9	32.0	38.3
1528117862	30x10	8.2	27.1	34.5	46.7
1528117864	30x10	504.4	26.8	28.0	30.2
1570986545	30x15	3470.5	51.7	62.4	99.1
1528117980	30x15	43200.0	52.4	71.8	120.0
1528117982	30x15	0.9	49.6	61.9	72.1
1528117984	30x15	43200.0	49.8	62.3	86.5
1528117986	30x15	407.6	49.5	64.9	85.7
1528118414	40x5	1.6	21.0	24.8	30.2
1528118418	40x5	1.1	22.3	30.7	42.0
1528118421	40x5	0.6	20.7	29.0	35.6
1528118423	40x5	0.9	23.6	31.2	51.0
1528118426	40x5	1.2	20.8	23.4	29.5
1528118631	40x10	43200.0	54.6	66.0	86.1
1528118633	40x10	4874.2	48.6	59.6	67.9
1528118635	40x10	437.0	47.0	47.3	47.8
1528118637	40x10	13.3	46.2	49.5	55.5
1528118638	40x10	997.2	47.1	53.0	73.5
1528118658	40x15	43200.0	83.6	121.2	170.2
1528118660	40x15	43200.0	85.0	106.9	124.4
1528118662	40x15	52.0	85.5	90.4	103.9
1528118663	40x15	107.0	94.2	108.6	150.4

1528118665	40x15	43200.0	89.5	117.7	147.1

From the tables we can observe that the proposed heuristic is able to produce high quality solutions with low variability, sometimes finding the optimum and doing it within a reasonable and predictable time frame that in some instances beats the model, especially on hard and larger instances.

5. Conclusions and Recommendations

The Capacitated Clustering Problem (CCP) is a difficult problem to solve that has many applications in diverse fields requiring design, distribution and pattern recognition. One of the most important fields tackling this problem is communications, specifically wireless communications in cellular networks. For this case, the CCP translates into a problem where all base stations must be assigned to one RNC, each base station contributes to the RNC the amount of traffic it's associated with and each RNC must not exceed the amount of traffic it is able to process. For this particular case, the CPP's objective function becomes maximizing the number of handovers between base stations assigned to the same RNC.

The CCP's model is not able to solve real life instances of the problem thus arising the need of developing a heuristic capable of giving solutions within a reasonable gap and time. The model is used to run small instances of the problem and for each record the number of handovers and time it takes to run. The goal is using those results to develop a heuristic capable of finding feasible solutions such that the average gap is always bellow 5%, meaning that it won't depend on instance size, and that the time is lower than $|R| * |T|/3$ seconds.

The proposed heuristic was a GRASP-Tabu consisting on loop running a 3-phase greedy for constructing the initial arrangement that may or may not be feasible. Then a Tabu search with two neighborhoods that guaranty a change in the solution structure: base station reassignment and two base station exchange improves the initial arrangement, also allowing for non-feasible arrangements during the iterations and penalizing said non-feasibility in a way that after some iterations the algorithm returns to a feasible region. After $|R|*|T|*4$ iterations without improvement the solution is returned, and another GRASP iteration begins. The GRASP-Tabu will not stop until 1000 iterations without improvement have passed.

A set of 40 instances were run through the model and the finished heuristic, the results showed two trends, the first is that average gap increases slightly as the size of the instance grows and the second shows that for instances where $|T|/|R| > 2$ the gap is lower than those where $|T|/|R| \leq 2$. The first trend is kind of obvious since the bigger the problem is the more possible arrangements there are but the second one shows that the more base stations there are per RNC the easiest the problem get. The results also showed that the highest average gap for an

individual instance was 34% and for its group (40x5) was 7%, meaning that the proposed GRASP-Tabu heuristic needs refinement to reach the stated goals.

The conclusion is that although the heuristic runs way faster than the model the gap is still too variable to be used on real life applications, thus for future work a third or more neighborhoods such as a 2-1, 2-2, and 3-2 base station exchange could be added to the Tabu search so that more possible solutions are available during the iterations. Another possible improvement could derive from using design of experiments for tuning the parameters used in the GRASP instead of letting them cycle through values according to the current iteration.

References

Ahmadi, S. and Osman, I. (2005) Greedy random adaptive memory programming search for the capacitated clustering problem. *Eur. J. Oper. Res. 162(1),* 30–44.

Brimberg, J., Mladenović, N., Todosijević, R., & Urošević, D. (2019). Solving the capacitated clustering problem with variable neighborhood search. *Annals of Operations Research*, *272*(1-2), 289-321.

Brucker, J. (1978). On the Complexity of Clustering Problem. *Lecture Notes in Economics and Mathematical Systems, vol. 157,* pp. 45–54. Springer, Berlin/Heidelberg.

Deng, Y., & Bard, J. F. (2011). A reactive GRASP with path relinking for capacitated clustering. *Journal of Heuristics*, *17*(2), 119-152.

Gallego, M., Laguna, M., Martí, R., & Duarte, A. (2013). Tabu search with strategic oscillation for the maximally diverse grouping problem. *Journal of the Operational Research Society*, *64*(5), 724-734.

Jabbari, B., Colombo, G., Nakajima, A., and Kulkarni, J. (1995). Network Issues for Wireless Communications. *IEEE Communications magazine*, pp. 88-98.

Martínez-Gavara, A., Campos, V., Gallego, M., Laguna, M., & Martí, R. (2015). Tabu search and GRASP for the capacitated clustering problem. *Computational Optimization and Applications*, *62*(2), 589-607.

Martinez-Gavara, A., Landa-Silva, D., Campos, V., & Marti, R. (2017). Randomized heuristics for the capacitated clustering problem. *Information Sciences*, *417*, 154-168.

Morán-Mirabal, L. F., González-Velarde, J. L., Resende, M. G., & Silva, R. M. (2013). Randomized heuristics for handover minimization in mobility networks. *Journal of Heuristics*, *19*(6), 845-880.

Morán-Mirabal, L.F., González-Velarde, J.L., and Resende, M.G.C. (2013). Automatic tuning of GRASP with evolutionary path-relinking. *In: Blesa, M.J. (ed.) Hybrid Metaheuristics, vol. 7919*, pp. 62–77. Lecture Notes in Computer Science, Heidelberg.

Mulvey, J. M., & Beck, M. P. (1984). Solving capacitated clustering problems. *European Journal of Operational Research*, *18*(3), 339-348.

Negreiros, M., & Palhano, A. (2006). The capacitated centred clustering problem. *Computers & operations research*, *33*(6), 1639-1663.

Tollis, I. G. (1996, June). Optimal partitioning of cellular networks. In *Proceedings of ICC/SUPERCOMM'96-International Conference on Communications* (Vol. 3, pp. 1377-1381). IEEE.

Zhou, Q., Benlic, U., Wu, Q., & Hao, J. K. (2019). Heuristic search to the capacitated clustering problem. *European Journal of Operational Research*, *273*(2), 464-487.