A 64-bit Linux operating system (Fedora distribution) was the operating system used. The message-passing implementation OpenMPI, version 1.6.1, was manually compiled with the distribution-supplied gcc compiler, version 4.4.4.

6.5.1 Test networks

The parallel performance of PRATO was tested with real radio networks of different sizes. In order to create the synthetic test data sets with an arbitrary number of transmitters, a group of 2,000 transmitters of a real network was used. The configuration parameters of these transmitters resembled those of the LTE network deployed in Slovenia by Telekom Slovenije, d.d., which were, in turn, randomly replicated and distributed over the whole target area. The path-loss predictions were calculated using the radio-propagation model introduced in Section 6.3.2. The DEM area, as well as the clutter data, covered 20,270 km^2 with a pixel resolution of 25 m^2. The clutter data contained different levels of signal loss due to land usage. For all the points within a radius of 20 km around each transmitter, a receiver positioned 1.5 m above the ground was assumed, and the frequency was set to 1,843 MHz.

6.5.2 Weak scalability

The weak-scalability experiments are meant to analyze the scalability of the parallel implementation in cases where the workload assigned to each process (one MPI process per processor core) remains constant as the number of processor cores is increased. It follows that the number of transmitters deployed over the target area is directly proportional to the number of processor cores and worker processes. This was accomplished by assigning a constant number of transmitters per core, while increasing the number of cores hosting the worker processes. Here, the following numbers of transmitters per worker/core were tested {5,10,20,40,80}, by progressively doubling the number of worker processes from 1 to 64.

Problems that are particularly well-suited for parallel computing exhibit computational costs that are linearly dependent on the size of the problem. This property, also referred to as algorithmic scalability, means that proportionally increasing both the problem size and the number of cores results in a roughly constant time to solution.

The master-worker (MW) configuration performs result aggregation continuously, i.e., while receiving the intermediate results from the worker processes. In contrast, the master-worker-DB (MWD) setup performs the result aggregation as the final step. This set of experiments is meant to investigate how the proposed MWD technique compares with the classic MW approach in terms of scalability when dealing with a constant computational load per core.

An important fact about the presented simulations when using multi-threaded implementations is to avoid oversubscribing a computing node. For example, if deploying four worker processes over a quad-core CPU, the extra threads will have a counter effect on the parallel efficiency, since the CPU resources would be exhausted, which slows the whole process down. For this reason, we have deployed three worker processes per computing node, leaving one core free for executing the extra threads.

6.5.2.1 Results

The results represent the best running time out of a set of 20 independent simulation runs, for which the transmitters and the rank ordering of the worker processes were randomly selected. The collected running times for the weak-scalability experiments are shown in Figure 6.10. All the measurements express wall-clock times in seconds for each setup and

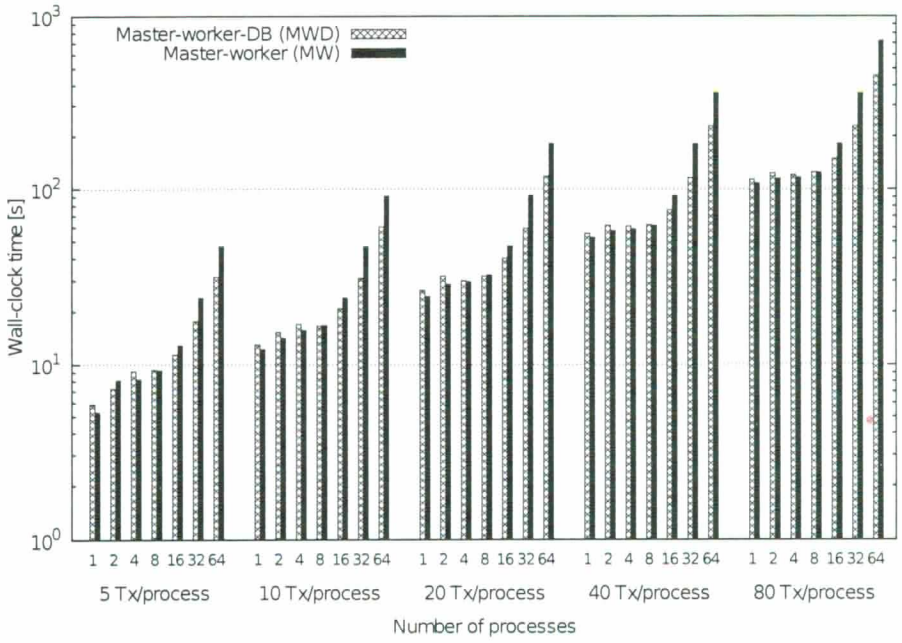

Figure 6.10: Measured wall-clock time for weak-scalability experiments, featuring MW and MWD setups. Experiments allocated one MPI worker process per core. The wall-clock time axis is expressed in a base-10 logarithmic scale, whereas the axis representing the number of cores is expressed in a base-2 logarithmic scale.

problem instance, defined as the number of transmitters per process (Tx/process). The wall-clock time represents the real time that elapses from the start of the master process to its end, including the time that passes while waiting for the resources to become available. The running-time improvements of the MWD versus the MW setup are shown in Table 6.1.

The time measurements observed from the weak-scalability results show that the classic MW approach performs well for up to four worker processes. When using eight worker processes, the MW setup is practically equivalent to the MWD approach, indicating that the master process is being fully exploited. When increasing the problem size and the number of worker processes to 16, the running-time gain is already clear, favoring the MWD configuration. This gain keeps growing, although slower, as we increase the number of worker processes to 32 and 64, confirming the hypothesis that in a classic MW approach, the parallel efficiency is bounded by the capacity of the master process to serve an increasing number of

Table 6.1: Running-time gain (in percent) of the simulations for the weak-scalability of the MWD setup relative to the classic MW approach.

	Number of cores						
TX/core	1	2	4	8	16	32	64
5	-11.39	-10.42	-11.14	-0.95	11.75	26.15	32.53
10	-5.84	-7.78	-7.67	0.91	12.81	33.28	33.55
20	-8.59	-10.88	-1.04	1.95	14.29	35.23	35.27
40	-5.26	-6.90	-3.68	-0.67	17.27	36.23	36.65
80	-5.29	-7.11	-3.20	-0.31	17.94	36.32	36.57

worker processes. Interestingly, the gain when using 32 and 64 worker processes is almost the same. After further investigation, the reason for this behavior was found: the new bottleneck was the LAN being completely saturated by the worker processes. Consequently, they have to wait for the network resources to become available before sending or receiving data, which is not the case when running the MW setup. Therefore, using the MWD approach a hardware constraint is hit, meaning that the bottleneck is no longer at the implementation level. Moreover, since the master process is far from overloaded when serving 64 worker processes, it can be expected that the MWD approach will keep scaling if a faster network infrastructure is used, e.g., 10-gigabit Ethernet or InfiniBand.

Certainly, the parallel version of PRATO scales better using the MWD approach, when challenged with a large number of transmitters (5,120 for the biggest instance) over 64 cores. This fact shows PRATO would be able to calculate the radio-coverage prediction for real networks in a feasible amount of time, since many operational radio networks have already deployed a comparable number of transmitters, e.g., the 3G network within the Greater London Authority area, in the UK [118]. For a more in-depth discussion and experimentation about real-world planning scenarios, see Chapter 10.

Not being able to achieve perfect weak scalability using the MWD setup is due to a number of factors. Specifically, the overhead time of the serial sections of the parallel process grows proportionally with the number of cores, e.g., aggregation of the intermediate results, although the total contribution of this overhead remains low for large problem sizes. Moreover, the communication overhead grows linearly with the number of cores used. Consequently, the findings of Huang et al. [82] can be confirmed, who concluded that the data-set size should be large enough for the communication overhead to be hidden by the calculation time. This ensures profitable parallelization in terms of running-time reduction.

6.5.3 Strong scalability

This set of simulations is meant to analyze the impact of increasing the number of computing cores for a given problem size, i.e., the number of transmitters deployed over the target area does not change, but only the number of worker processes used is increased. Here, the following number of transmitters were tested {1,280, 2,560, 5,120}, by gradually doubling the number of workers from 1 to 64 for each problem size.

6.5.3.1 Results

Similar to the weak-scalability experiments, these time measurements show that when applying a classic MW approach, the running-time reduction starts flattening when more than eight worker processes were used. Moreover, the running times for 16, 32 and 64 worker processes are the same, i.e., they do not improve due to the master process being saturated. In contrast, when using the proposed MWD technique, the running-time reduction improves for up to 32 worker processes, after which there is no further improvement since the network was being fully exploited. These results clearly show that when applying parallelization using a larger number of worker processes, the master process becomes the bottleneck of the MW approach. When using the MWD configuration, a steady running-time reduction is observed, until a hardware constraint is hit, e.g., the network infrastructure.

The overhead of sending/receiving asynchronous messages in order to support heterogeneous systems was also measured. It was found that this overhead never exceeds 0.02% of the total running time for the MW experiments, and 0.01% for the MWD experimental set.

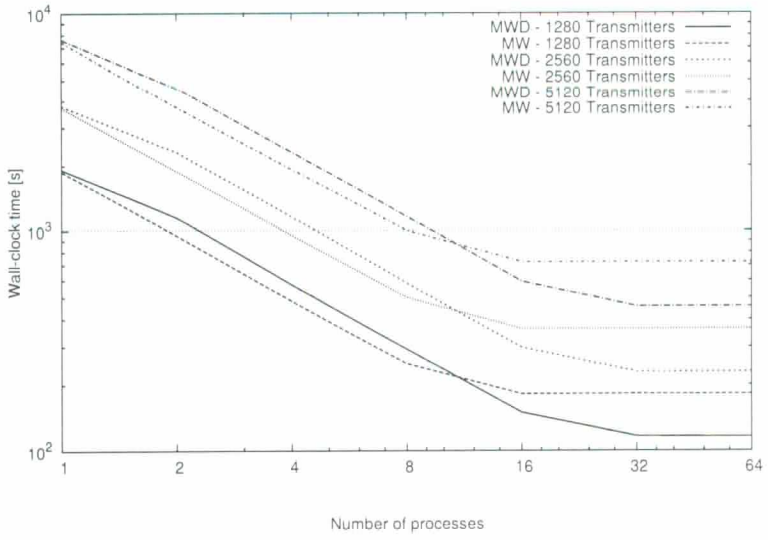

Figure 6.11: Measured wall-clock time for strong-scalability experiments, featuring MW and MWD setups. Experiments assigned one MPI worker process per core. The wall-clock time axis is expressed in a base-10 logarithmic scale, whereas the axis representing the number of cores is expressed in a base-2 logarithmic scale.

6.5.3.2 Speedup

In order to further analyze how well the PRATO scales using the MW and MWD approaches, the performance of the parallel implementation in terms of its speedup was measured, which is defined as:

$$S(NP) = \frac{execution\,time\,for\,base\,case}{execution\,time\,for\,NP\,cores}, \tag{6.5}$$

where NP is the number of cores executing the worker processes. The parallel implementation running on only one core was the base case for comparisons. The serial implementation is not a good base comparison for the parallel results as it does not reuse the resources between each transmitter-coverage calculation and it does not overlap the I/O operations with the transmitter computations. In practice, this means that several concatenated runs of the serial version would be considerably slower than the single-worker configuration.

Using the speedup metric, linear scaling is achieved when the obtained speedup is equal to the total number of processors used. However, it should be noted that a perfect speedup is almost never achieved, due to the existence of serial stages within an algorithm and the communication overhead of the parallel implementation.

Figure 6.12 shows the average speedup of the parallel implementation for up to 64 worker processes, using the standard MW method and the proposed MWD approach. The average speedup was calculated for the three different problem instances, i.e., 1,280, 2,560, and 5,120 transmitters deployed over the target area. The number of transmitters used in these problem sizes is comparable to several real-world radio networks that were already deployed in England, e.g., Hampshire County with 227 BSs, West Midlands with 414 BSs, and Greater London Authority with 1,086 BSs [118]. Note that it is common for a single base station to host multiple transmitters.

The plotted average speedup clearly shows the minimal overhead of the MWD approach when using a small number of worker processes. This overhead accounts for the final aggregation of the intermediate results at the DB, which in the MW configuration is performed

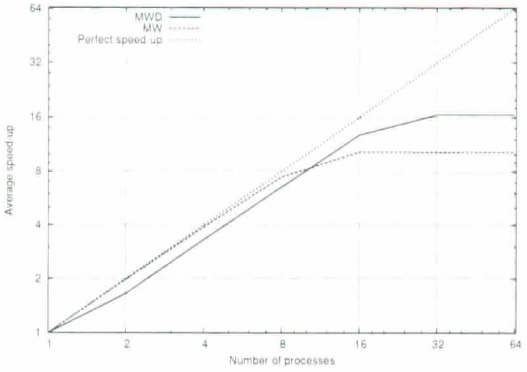

Figure 6.12: Average speedup for strong-scalability experiments. Both axes are expressed in a base-2 logarithmic scale.

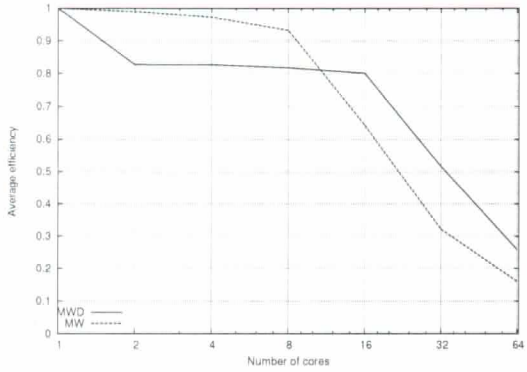

Figure 6.13: Average parallel efficiency for strong-scalability experiments. The parallel-efficiency axis is expressed in a linear scale, whereas the axis representing the number of cores is expressed in a base-2 logarithmic scale.

along worker processing. Like before, the DB component allows the parallel implementation to fully exploit the available computing resources when deploying a larger number of worker processes, until the network-capacity limit is met. Of course, these results are directly correlated with the wall-clock times shown in Figure 6.11.

6.5.3.3 Efficiency

Another measure to study how well PRATO utilizes the available computing resources considers the parallel efficiency of the implementation. The definition of parallel efficiency is as follows:

$$E(NP) = \frac{S(NP)}{NP}, \tag{6.6}$$

where $S(NP)$ is the speedup as defined in Equation (6.5), and NP is the number of cores executing worker processes. Figure 6.13 shows the average parallel efficiency of the parallel implementation for different problem sizes, as the number of processing cores was increased. Like for the speedup measure, the average parallel efficiency from the same problem instances was calculated.

The ideal case for a parallel application would be to utilize all the available computing resources, in which case the parallel efficiency would always be equal to one as the core count increases. From the plot in Figure 6.13, it can be observed that the efficiency of the MWD approach is better than in the MW case for larger number of processes, and as long as there was available capacity at the LAN level. In accordance to the previous analysis, the under utilization of the computing resources is more significant when the master process is overloaded (in the MW case) than when the network infrastructure is saturated (in the MWD case). The lower efficiency is directly proportional to the number of idle worker processes that wait either for the master process (MW case) or for network access (MWD case).

Overall, the experimental results confirm that the objective of fully exploiting the available hardware resources is accomplished when applying the presented MWD approach, thus improving the scalability and efficiency of PRATO when compared to a traditional MW technique.

6.5.4 Performance analysis

<Experiments not yet finished ...???>

MISSING?

6.6 Summary

PRATO, a parallel radio-coverage prediction tool for radio networks, has been presented in this chapter. The tool is intended to be used for radio-network planning analysis and decision support. Its high-performance capabilities make it ideal for automatic-optimization tasks that require a large number of evaluations.

The parallel implementaion of PRATO includes a novel parallel technique for master-worker configurations. The introduced MWD technique, which combines the use of a DB system with a work-pool approach, delivers improved performance when compared with a traditional MW setup. Moreover, the presented system provides parallel and asynchronous computation that is completely independent of the GIS used, in this case the GRASS environment. Consequently, a GIS installation is only needed on the master node, thus simplifying the required system setup and greatly enhancing the applicability of this methodology in different environments.

The extensive simulations, performed on the DEGIMA cluster of the Nagasaki Advanced Computing Center, were analyzed to determine the level of scalability of the implementation, as well as the impact of the presented methods for parallel-algorithm design aimed at spatial-data processing. The conducted analyses show that when using the MWD approach, PRATO is able to calculate the radio-coverage prediction of real-world radio networks in a reduced amount of time. Moreover, the experimental results show that PRATO has a better scalability when using the MWD approach than the standard MW setup, since it is able to completely saturate the network infrastructure of the computer cluster. These promising results also show the great potential of the MWD approach for parallelizing different time-consuming tasks dealing with spatial data, where DBs form an intrinsic part of almost all GIS. Furthermore, the automatic optimization of radio networks, where a large number of radio-propagation predictions take part in the evaluation step of the optimization process, can also an benefit from the improved performance of PRATO. Indeed, this last point will be further discussed and validated in the following chapters.

The performance of the worker processes has been additionally improved by including the implementation of the radio-propagation algorithm on GPU. The use of GPU hardware is optional, i.e., it is exploited only if it is available on the computing nodes that host the worker processes. The experimental simulations showed a significant speedup due to the GPU implementation, when ~~compared to~~ the CPU-only version.

To the best of the author's knowledge, neither the MWD parallel technique nor the parallel implementation of the radio-prediction algorithm as presented in this chapter, have yet been described in the related literature.

WHERE?!?

WHY IS THIS NOT SHOWN?

7.1 Motivation

Solving the service-coverage problem for radio networks has received a great deal of attention in the past years. Its complexity demands the confluence of different skills in areas such as propagation of radio signals, telecommunications and information systems, among others.

Even several decades after the launch of the first commercial GSM network, service-coverage planning remains a key problem that all mobile operators have to deal with. Its intricacy arises from the wide range of different combinations of configuration parameters and their evaluation-time complexity. One crucial parameter, which is mainly subject of adjustment, is the transmit power of the pilot signal (see Section 3.2, ~~Chapter~~ 3).

Regardless of the mobile technology used, e.g., GSM, UMTS or LTE, a lower transmit power generates less interference, which, in turn, translates into more capacity of the radio link (see Chapter 3). Moreover, reducing pilot-power usage is also related to issues regarding the human exposure to the electromagnetic fields generated by BS antennas [50]. During the past few years, public opinion has been extremely sensitive regarding this issue, and thus many countries have already imposed safety standards to limit the electromagnetic field levels produced by antennas in a given range.

From the UMTS perspective, minimizing pilot-power usage leaves more power available for increased network capacity. This is especially important if the traffic and other channels are configured relative to the pilot channel [79]. Moreover, as the demand for internet access and data services increases [41], so does the pressure on existing network infrastructure, making parameter optimization the only viable solution in the short-term [111].

The idea of using autonomous agents for optimization is not new. It has proven to be a solid optimization approach for solving different types of problems, not only within the area of radio networks [34, 50], but also in other fields [166, 169]. The increased computational-time complexity when dealing with big problem instances is tackled using a parallel, agent-based algorithm on GPU. This minimizes the overhead when deploying a larger number of agents working in parallel over the service area, only limited by the amount of memory available.

7.2 Related work

There are several approaches in the literature that address the service-coverage problem in radio networks [9, 111, 142]. Some of them even claim to achieve near-optimal solutions [148]. As a matter of fact, most formulations are only useful for small network instances and often fail when challenged with larger, real-world networks.

A genetic-algorithm approach for solving the service-coverage problem for GSM networks was presented in [100]. The proposed solution is based on the physical distribution of BSs in order to maximize coverage. The simulations were performed on a test network with 40 candidate sites for BS antennas.

In [148], Siomina and Yuan considered the problem of minimizing the total amount of pilot power for UMTS networks, subject to a full coverage constraint. They tackled the problem with an iterative linear-programming approach, reporting very good results for some test networks, containing from 15 to 65 base stations. The authors noted that bigger problem instances could not be solved because of hardware constraints on the target platform.

As for LTE networks, the service-coverage problem was addressed in [162]. The authors presented an algorithm, based on reinforcement learning, to tackle three aspects of the coverage problem, i.e., coverage holes, weak coverage and pilot pollution. The experimental simulations, performed on 3 BSs, used different antenna-tilt configurations as the proposed solutions.

7 The service-coverage problem

The high-performance of PRATO, the radio-coverage simulation framework presented in Chapter 6, allows dealing with big problem instances in a reduced amount of time. Additionally, it enables tackling optimization problems that, because of their size, are out-of-reach of traditional approaches, mainly due to the computational-time complexity of their objective-function evaluation.

In this chapter, the challenge is to exploit PRATO for solving one the of classic optimization problems of radio networks: the service-coverage problem. Considering the minimization of the total amount of pilot power subject to a full coverage constraint, a novel optimization approach is introduced. The presented method, based on parallel autonomous agents, gives very good solutions to the problem in an acceptable amount of time. The parallel implementation takes full advantage of GPU hardware in order to achieve considerable speed-up. The analysis of the experimental results, considering six real-world, radio networks of different sizes, studies solution-quality and performance aspects.

The content of this chapter extends the research work published by the author in [18] and [20]. The rest of this chapter is organized as follows. Section 7.1 gives a description of the coverage problem and its motivation from the mobile operator's perspective. In Section 7.2, a short overview of related research works is given, before formally introducing the key elements of the service-coverage problem in Section 7.3. The parallel-agent approach, as well as the strategies used for result comparison, are presented in Section 7.5, followed by the simulations and their analyses in Section 7.6.

ŠE NE SPOMNIM, SI TO IMEL PRI VSAKEM POGLAVJU??

The service-coverage problem, as presented in this chapter, corresponds to achieving full coverage of the target area, without coverage holes.

7.3 Radio-network model

Extending the representation of a radio-network model from [112], this section presents the definitions of all the elements included in the mathematical model used for the simulations.

The goal here is to analyze the state of the network in a given situation, i.e., a 'snapshot' at an arbitrary instance. A snapshot consists of a set of UEs having individual properties, such as location, and equipment type. The static approach inherently ignores dynamic effects that influence the system, like fast power control.

For additional information regarding mathematical models of comparable problems, see [111].

7.3.1 Basic elements

Consider a UMTS network with a set of antenna installations (cells), C. A RSG of a given resolution represents a geographical area, A_{total}, within which a set of UEs, M, is spatially distributed over the pixels of A_{total}. Further, $L^{\downarrow}_{cm}$ is defined as the downlink attenuation factor between cell $c \in C$ and UE $m \in M$. Similarly, $L^{\uparrow}_{mc}$ represents the uplink attenuation factor between UE m and cell c. The attenuation factor values are calculated by performing signal-propagation predictions for every pair (c, m), $c \in C, m \in M$, using the radio-propagation model introduced in Section 6.3.2, Chapter 6. These predictions already include losses and gains from cabling, hardware, and user equipment.

The amount of power allocated to the pilot signal of cell c is denoted as p_c, and it can adopt any value from the sorted set of available pilot power levels, $P_c = \{p_c^1, p_c^2, ..., p_c^k\}$, where p_c^k is the maximum power.

Based on the introduced elements, the received pilot power from cell c to UE m is $L^{\downarrow}_{cm} p_c$.

7.3.2 Coverage

An UE m within the area A_{covered} is under service coverage if at least one cell c covers it. Cell coverage is provided to an UE m from a cell c if its signal-to-interference ratio, $sir(c, m)$, at the RSG pixel where m is located, is not lower than a given threshold, γ^{cov}:

$$sir(c,m) = \frac{L^{\downarrow}_{cm} p_c}{\tau_0 + \sum_{i \in C} L^{\downarrow}_{im} p_i} \geq \gamma^{\text{cov}}, \tag{7.1}$$

where τ_0 is the thermal noise. For convenience, a binary function is defined to determine the coverage of a UE m by a cell c. So, for any pair (c, m), $c \in C, m \in M$, the coverage of UE m by cell c is defined as:

$$cov(c,m) = \begin{cases} 1 & if\ sir(c,m) \geq \gamma^{\text{cov}} \\ 0 & otherwise \end{cases}. \tag{7.2}$$

A set, denoted as C_m, $C_m \subset C$, contains all the cells covering an UE m. From this set, the cell with the highest $sir(c, m)$ is refered to as the best server, and denoted as c^*_m.

Notice that the described radio-network model is easily adaptable for different mobile technologies, e.g., GSM, UMTS and LTE. For example, if solving the service-coverage problem for UMTS, it would be reasonable to assume that all cells in the network operate at maximum power, and adapt Equation (7.1) accordingly. This is, from the interference point of view,

the worst-case scenario [30, 148]. This assumption guarantees, that even under heavy user traffic, full coverage of the service area is maintained, due to the cell-breathing principle [79].

7.4 Problem definition

In the problem of optimization of pilot powers for service coverage, the objective is to find a set of pilot-power settings for all cells in the network, such that the total pilot power used is minimized, and a given service coverage criteria is fulfilled. In other words, solving the service-coverage problem corresponds to finding the pilot power levels p_c, for all cells $c \in C$, such that coverage of at least b UEs is guaranteed, while the total amount of pilot power used is minimized. Here, full coverage of the service area is being considered, thus $b = |M|$. Consequently, the optimization objective is defined as follows:

$$P^* = \min \sum_{c \in C} p_c, \tag{7.3}$$

subject to

$$\frac{\sum_{m \in M} cov(c_m^*, m)}{b} = 1. \tag{7.4}$$

It has been proved that the problem of pilot-power optimization for full coverage of the service area is NP-hard, since it can be reduced to the set-covering problem [168]. Consequently, as long as $P \neq NP$, it is unfeasible that a polynomial-time algorithm exists, which is able to find an exact solution to this problem.

7.5 Optimization approaches

Since some of the analyzed problem instances are part of a real mobile network deployed in Slovenia by Telekom Slovenije, d.d., there are no references in the literature of other optimization techniques dealing with exactly the same data set. For this reason, two different strategies for setting the pilot power are being presented. They should provide a basis for the comparison of the experimental results. The first strategy is the attenuation-based pilot power, presented in [142], in which a pixel of the service area is always covered by the cell with the maximum attenuation-factor value, i.e., the minimum path loss. The second strategy is the presented parallel-agent approach, based on ideas inspired by two-dimensional cellular automata [135] and metaheuristics [159]. A detailed description is given in Section 7.5.2.

Similar criteria for result comparison have also been used in [142, 148].

7.5.1 Attenuation-based approach

The first heuristic for setting the pilot power of all cells in the network is known as attenuation-based, since it relies on the downlink-attenuation factor, $L_{cm}^{\downarrow}$. An UE located on some pixel of the service area is always covered by the cell with the maximum $L_{cm}^{\downarrow}$. Whenever the maximum available power, p_c^k, is the same for all the cells in the network, this is equivalent to selecting the cell with the minimum required pilot power to cover a UE m. Hence, under this assumption, the cell c covering UE m is identified as:

$$p_{cm}^{\text{att}} = \min p_c \, \forall c \in C \iff cov(c, m) = 1 \tag{7.5}$$

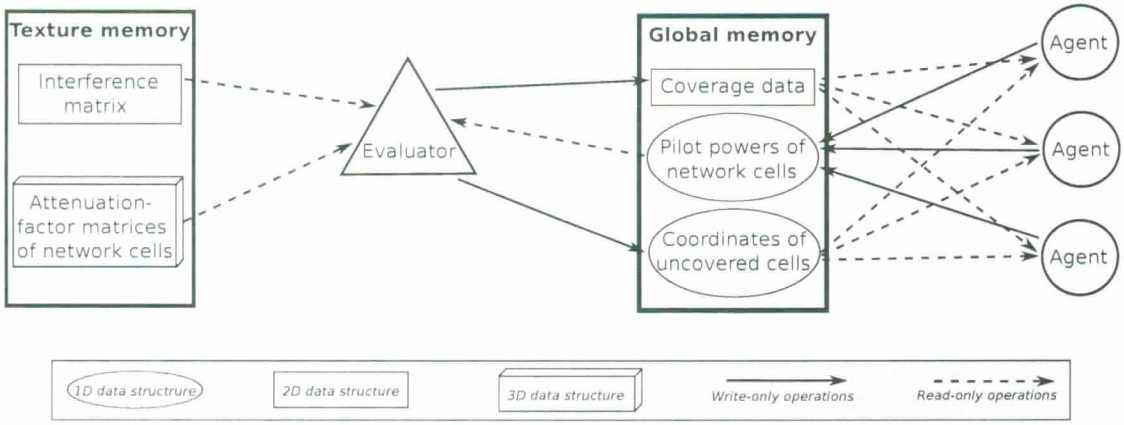

Figure 7.1: Architecture of the parallel, agent-based optimization system on GPU.

Picking the cells conforming to Equation (7.5) and setting the pilot powers accordingly, full coverage of the service area is achieved. The solution exhibits a total pilot power defined as:

$$P^{\text{att}} = \sum_{c \in C} \max p_{cm}^{\text{att}}. \tag{7.6}$$

The procedure to find a cell c for every UE m in the service area consists in sorting, in descending order, all UEs by their attenuation-factor values, $L_{cm}^{\downarrow}$. The solution is thus established by the first b UEs of the sorted sequence, taking the maximum pilot-power setting for a cell into account, i.e., p_{cm}^{att}.

7.5.2 Parallel-agent approach

In the parallel-agent approach, a set of autonomous worker agents explore the target geographical area, A_{total}, in order to optimize the pilot-power consumption. Each agent randomly moves over the A_{total} as it dictates different changes to the pilot power of the cells. PRATO, the radio-coverage framework presented in Chapter 6, performs the objective-function evaluations and radio-propagation predictions based on the proposed changes of the agents.

The moving process during the optimization is strictly random. However, several physical properties that are exclusive to the service-coverage problem are being exploited during the exploration of the search space. Additionally, whenever the current solution breaks any of the given constraints, the optimization process is guided back to the space of valid solutions, providing a mechanism for improving exploration and escaping from local optima.

Because the behaviour of the agents is independent between each other, a parallel implementation is fairly straight-forward to achieve. Figure 7.1 shows the architecture of the agent-optimization system. In this GPU-only architecture, agents work in a parallel and autonomous manner, while the evaluator reacts to their changes.

7.5.2.1 Objective-function evaluation

The evaluator represents a central component of the optimization system. It reacts to the pilot-power changes by recalculating the objective-function value. Recall that the objective-function evaluation involves the radio-coverage prediction of the service area and the calculation of the total pilot power used by the cells in the target network.

After a short initialization, during which the attenuation-factor matrices of all the cells and the interference matrix are calculated, the evaluator computes the coverage of the service area based on the pilot powers supplied as the initial solution. Initial solutions are randomly generated from valid pilot-power settings that conform to the full coverage constraint.

The evaluator also maintains a special part of the memory (see "Coordinates of uncovered cells" in Figure 7.1) that is intended for registering uncovered areas, i.e., $\overline{A_{\text{covered}}}$. If Equation (7.4) does not hold, the "special" agents randomly select a location from this portion of memory so that a valid solution may be reached again.

It is worth mentioning that the evaluator itself has no influence in the optimization process from a quality point-of-view. Its task is to provide feedback and updated information to the agents that move through the service area. From a performance point-of-view, the importance of the evaluator is significant, as it will be shown in the following sections.

The evaluation of the objective function was completely implemented on the GPU using OpenCL (see Section 5.2, Chapter 5). The reason behind this decision is the impact objective-function evaluation has on the performance of the optimization system as a whole, as discussed in Section 2.4, Chapter 2. The implementation of the agents is also based on the GPU, which drastically reduces the number of data transfers between CPU and GPU, since all problem elements are available on the GPU during the optimization process. Consequently, careful memory utilization and organization are critical to successfully accommodate all involved problem elements on the GPU, the memory of which is significantly smaller than the RAM memory available in desktop computers.

7.5.2.2 Autonomous agents

The agents apply the pilot-power changes only considering local information. Each of them encapsulates a set of steps that is consistently applied as it randomly moves through the service area of the network. Whenever an agent arrives at a new location, the set of covering cells is calculated, i.e., C_m.

The step set an agent applies following this point is directly related to $|C_m|$, whereas its movement is determined by $|\overline{A_{\text{covered}}}|$, i.e., the area without service coverage.

The behavior of an agent is dictated by the pseudo-code shown in Algorithm 7.1. The first four steps are responsible for guiding its movements. The coordinates are randomly selected from two sets, A_{total} and $\overline{A_{\text{covered}}}$. Only "special" agents may move to a location without service coverage, and they apply the step set SS_0 for as long as the solution is not valid. The portion of "special" agents used for correcting a solution is a parameter of the optimization process. During the following steps of Algorithm 7.1, the agent applies step sets SS_0 and SS_1 based on the number of cells in C_m.

If the current location of the agent is not covered by any cell, i.e., $|C_m| = 0$, the step set SS_0 is applied (see Algorithm 7.2). At the beggining, the cell with the highest attenuation factor that may cover a UE at this location, c', is selected. If several cells have the same $L^{\downarrow}_{cm}$ value, one of them is randomly chosen. Once c' is uniquely identified, the agent changes its pilot power by inc_rate dB.

The step set SS_1, the pseudo-code of which is listed in Algorithm 7.3, is applied if the location of the agent is under the coverage of one or more cells, i.e., $|C_m| \geq 1$. The first step randomly selects a cell from the set C_m, followed by a decrease of the pilot power of cell c'. This practice keeps the coverage constraint valid over A_{total}, although it might potentially break it on other areas. Ideally, every pixel of the geographical area has to be covered by exactly one network cell, although this is just a representation of a perfect solution that is unreachable because of irregularities in the network topology and the terrain.

Algorithm 7.1 Pseudo-code representing the behaviour of an agent.

```
repeat
    if is_special_agent() and |A_covered| > 0 then
        l ← pick_random_location(A_covered)
    else
        l ← pick_random_location(A_total)
    end if
    move(l)
    if |C_m| = 0 then
        apply(SS_0)
    else
        if |C_m| ≥ 1 then
            apply(SS_1)
        end if
    end if
until stopping_criterion()
```

Algorithm 7.2 Pseudo-code representing the step set SS_0, which is applied by the agents in areas with no service coverage.

```
repeat
    c' ← next_cell_with_max_att(m)
    p_c' ← adjust_power(c', inc_rate)
until p_c' ∈ P_c'
```

Algorithm 7.3 Pseudo-code representing the step set SS_1, which is applied by the agents in areas with service coverage.

```
repeat
    c' ← pick_random_cell(C_m)
    p_c' ← adjust_power(c', dec_rate)
until p_c' ∈ P_c'
```

In both step sets, SS_0 and SS_1, the agent makes sure that the new pilot power setting, $p_{c'}$, is an element of $P_{c'}$. If this is not the case, cell c' is discarded and another cell is repeatedly selected at the beggining of both step sets, until this condition is satisfied.

The values inc_rate and dec_rate are configurable parameters that should be set before starting the optimization process. They indicate the relative adjustment (expressed in dB) of the pilot power of cell c'. On the one hand, lowering the pilot power of a cell decreases the interference it creates within its coverage area and those of their neighbours. Since the $sir(c, m)$ value increases with lower interference, the coverage of m may be achieved by a neighbor cell with the same or lower pilot power. On the other hand, increasing the pilot power of the cell with the maximum attenuation factor improves coverage by evenly distributing the power among different network cells. This cell is, on average, the nearest one to the location of a UE m.

With the objective-function evaluation running on the GPU, a new performance bottleneck appeared. The limitation factor in this case was the CPU-to-GPU data transfers that occurred in each iteration of the optimization process (see Section 5.1, Chapter 5).

The GPU kernel of the agents is launched as one thread block that contains one thread per deployed agent. The thread block is organized in a one-dimensional grid. The initial location of each agent is randomly generated using the current system time as a random seed. Since OpenCL provides no function for random-number generation, a simplified version of Marsaglia's generator [104] was implemented.

The analysis each agent does about the received signals at the current location is saved into the shared memory of the thread block. It contains the network cell and its pilot-power setting. Since both numbers are of type *short*, there is enough space in a 16 KB shared-memory block to allocate 4,096 agents. The last step involves saving the new pilot powers into global memory. This step is performed by only one of the threads within the thread block in order to avoid memory-access conflicts. Updated pilot powers are saved in negative form to indicate that coverage re-calculation is needed for these cells. In case there are several updated pilot powers for one network cell, the median is calculated and applied as the new pilot power.

Even though coalesced access is not achieved by the GPU kernel of the agents, its sole implementation provided enhanced performance. This performance gain appears because of the lower number of data transfers between the CPU and the GPU, since most data are available in global memory. Moreover, the GPU kernel also produces the truly parallel behavior of the agents, as they all apply the pilot-power changes at the same time.

7.6 Simulations

7.6.1 Test networks

The test networks, Net_1, Net_2 and Net_3, are subsets of the real radio network deployed by Telekom Slovenije, d.d. The path-loss predictions were calculated using the radio-propagation model presented in Section 6.3.2, Chapter 6. A DEM with a 25 m^2 resolution was used as the terrain-profile data. The requirements for the coverage threshold, γ^{cov}, were provided by experts of the Radio Network department of Telekom Slovenije, d.d.

Net_1 is deployed over a densely populated urban area. For this reason, the value of γ^{cov} is lower here, since network capacity is the dominating factor, whereas coverage is flexible because of a larger cell density, i.e., more BSs per surface unit. Net_2 represents a network deployed over a rural area, meaning that the network capacity can be reduced at the cost of a better coverage, since the user density is lower. The last network, Net_3, represents a

Table 7.1: Sizes of the test networks used for experimentation of the service-coverage problem, in terms of equipment and geographical area.

	Number of base stations	Number of cells	Surface [km^2]	Resolution [m^2]
Net_1	26	77	100.00	25
Net_2	8	23	306.25	25
Net_3	45	129	405.00	25
Net_4	65	193	56.25	50
Net_5	12	36	16.00	50
Net_6	50	148	56.25	50

Table 7.2: Network parameters of the test networks used for the service-coverage problem.

	p_c^k	τ_0	γ^{cov}
Net_1	15.00 W	$1.55{\cdot}10^{-14}$ W	0.010
Net_2	19.95 W	$1.55{\cdot}10^{-14}$ W	0.020
Net_3	15.00 W	$1.55{\cdot}10^{-14}$ W	0.015
Net_4	19.95 W	$1.55{\cdot}10^{-14}$ W	0.010
Net_5	19.95 W	$1.55{\cdot}10^{-14}$ W	0.010
Net_6	19.95 W	$1.55{\cdot}10^{-14}$ W	0.010

suburban area with a densely populated, but relatively small, downtown center, where a compromise between the network capacity and the coverage has to be achieved.

The second group of test networks, including Net_4, Net_5 and Net_6, is part of the publicly available MOMENTUM project [85]. Test network Net_4 represents the city of Berlin (Germany), Net_5 represents the city of The Hague (Netherlands), and Net_6 is the largest network optimized in [148], representing a reduced version of Net_4. All networks include information about BS locations, path-loss predictions and realistic antennas, which are part of the scenarios provided by the MOMENTUM project.

Network configurations, that represent what could be an initial-network setup by common planning standards [79], were produced using the attenuation-based approach. Such configurations can be easily calculated by a network planner. Table 7.1 lists the number of BSs and cells per test network, as well as the size of the geographical area. Different network-parameter values used during the simulations are shown in Table 7.2.

7.6.2 Parameter settings of the parallel-agent approach

The parameter settings for the optimization algorithm were determined after some experimentation with the test networks. The parameter settings for each test networks are listed in Table 7.3.

Using a higher *inc_rate* than *dec_rate* reflects the behavior of the agents when full coverage of the service area is not guaranteed. In practice, areas without service coverage usually appear as irregular islands. The stopping criteria were set by limiting the total number of pilot-power changes an agent is allowed to make. The value was set to 10,000, even though for some of the test networks the best solutions were found in the first quarter of the experiment.

Table 7.3: Parameter settings of the parallel-agent approach for each test network.

	Agents	*inc_rate* (dB)	*dec_rate* (dB)	Pilot-power changes
Net_1	16	0.2	-0.1	10,000
Net_2	16	0.2	-0.1	10,000
Net_3	16	0.2	-0.1	10,000
Net_4	6	1.0	-0.1	10,000
Net_5	2	1.0	-0.1	10,000
Net_6	6	1.0	-0.1	10,000

Table 7.4: Optimization results after applying two different approaches for solving the service-coverage problem. All values are expressed in Watts.

	Attenuation-based		Parallel agents	
	Total power	Average pilot power	Total power	Average pilot power
Net_1	419.292	5.445	137.064	1.780
Net_2	78.297	3.404	33.344	1.450
Net_3	1,014.113	7.861	582.954	4.519
Net_4	179.876	0.932	145.715	0.755
Net_5	73.872	2.052	34.884	0.969
Net_6	147.014	0.993	112.332	0.759

7.6.3 Experimental environment

All experiments were performed on a multi-core Intel i7 2.67 GHz desktop computer with 6 GB of RAM running a 64-bit Linux operating system. The GPU hardware used was an ATI HD5570 with 1 GB of DDR3 RAM. The implementation language used was C, combined with OpenCL and OpenMPI extensions.

7.6.4 Results

The results achieved by the parallel-agent approach, listed in Table 7.4, improved the optimization objective significantly. They show that the pilot-power usage was reduced in all networks while the service area was kept under full coverage. Moreover, the parallel-agent solution for Net_1 improved the attenuation-based setting by more than 300%. As for Net_2, the observed improvement is around 232%, while the improvement for Net_3 is more than 170%.

As mentioned in Section 7.1, the practical interpretation of these results is directly related to a capacity increase due to reduced interference. Specifically, the capacity of the network has been significantly increased in all three problem instances. Therefore, a greater number of users should be able to access services provided by the mobile network, since coverage is assured. Moreover, an increased speed in high-speed data services should also be observed [79].

The last test network, Net_6, is the same as in [148]. When comparing these results to those of [148], an improvement of almost 3% can be observed in the solution provided by the parallel-agent approach.

VAPIŠI KATERIH VREDNOSTI SO DOSEGLI, DA SI PRIŠEL DO 3% ?