Chapter 4

Enhanced Multi-User Greedy Loading for DSM

In the previous chapter, a new algorithm called MIPB was presented which was shown to give a fair spectrum allocation that was close to the equal line weight optimum for OSB in a very short time and is scalable up to bundles of practical size. Unfortunately, a method to calculate arbitrary data rate points with MIPB was not developed.

This chapter details enhancements to the greedy algorithm [38] [39] [3] in order to calculate arbitrary data rate points in the rate regions. In particular, this chapter focuses on methods for improving the speed of operation of this algorithm and efficient parallelisation strategies. It also shows that the rate region performance of this algorithm is as close to optimal in realistic scenarios as to be of little practical difference.

4.1 Rate Regions

In this section, two methods for determining the rate regions using the greedy algorithm are presented and simulation results shown for various DSL networks.

In [3] the authors presented the multiuser greedy loading algorithm. The authors also briefly mentioned a method for calculating different data rates points using this algorithm. This method is shown in Algorithm 14.

Algorithm 14 Original Multiuser Greedy Rate finding algorithm [3]

repeat
  Execute Greedy Algorithm ▷ As in Algorithm 8
  for $n = 1 \ldots N$ **do**
    if $R_n < R_n^{target}$ **then**
      $w_n = w_n / \zeta$
    else $R_n > R_n^{target}$
      $w_n = w_n * \zeta$
    end if
  end for
until Rate Targets Met

Unfortunately, this method will lead to oscillations about the rate targets in most cases, unless the value of ζ is very small, which will in turn lead to very slow convergence behaviour. It is also not possible to know, a-priori, a value of ζ that will give good convergence behaviour.

Figure 4.1 shows the relationship between the value of ζ and the algorithm execution time for a 4 User ADSL2+ downstream scenario with the rate targets from Appendix A.1. The discontinuities in the graph are points where this algorithm does not converge.

4.1.1 Bisection of Rate Targets

By observing that the data rate on line n is a monotonic function of the value of w_n, it is proposed to utilise a bisection method to find the values of w_n on each line required to meet the data rate targets.

The bisection algorithm updates one value of w_n at a time and continues until all of the rate targets are within the specified tolerance. For each user, the greedy algorithm is executed to discover if the current value of w_n produces a rate above or below the rate target for that line. Having discovered this, the value w_n is either increased or decreased until a value on the opposite side of the rate target is found. If the rate target has not yet converged, the algorithm then proceeds to a traditional bisection.

In all of the examples used in this chapter, rate targets are set on all but one line, whose rate will therefore be maximised. The algorithm is illustrated in

Algorithm 15 Weight Bisection for Greedy Algorithm

$w_n = 1, \forall n$
repeat
  for $n = 1 \dots N$ **do**
    Execute Greedy Algorithm
    if $R_n < R_n^{target}$ **then** ▷ Found initial max
      $max = w_n$
      $w_n / = 2$
      repeat
        Execute Greedy Algorithm
        if $R_n < R_n^{target}$ **then**
          $max = w_n$ ▷ New max
          $w_n / = 2$
        else$R_n > R_n^{target}$
          $min = w_n$ ▷ Min found
        end if
      until min found
    else$R_n > R_n^{target}$ ▷ Found initial min
      $min = w_n$
      $w_n * = 2$
      repeat
        Execute Greedy Algorithm
        if $R_n < R_n^{target}$ **then**
          $max = w_n$ ▷ Max found
        else$R_n > R_n^{target}$
          $min = w_n$ ▷ New min
          $w_n * = 2$
        end if
      until max found
    end if
    $w_n = \frac{max+min}{2}$ ▷ Commence bisection between max and min
    repeat
      Execute Greedy Algorithm
      if $R_n < R_n^{target}$ **then**
        $max = w_n$
      else$R_n > R_n^{target}$
        $min = w_n$
      end if
      $w_n = \frac{max+min}{2}$
    until $|R_n - R_n^{target}| < R^{tol}$
  end for
until All Rate Targets Met

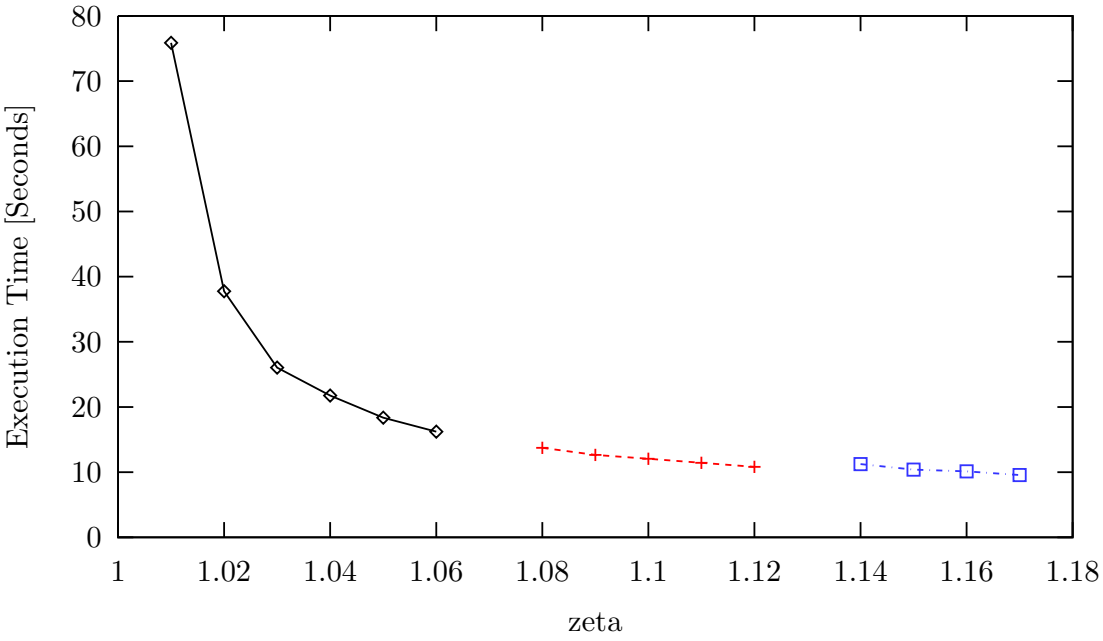


Figure 4.1: Relationship between execution times and the value of ζ used in Algorithm 14

Algorithm 15 and is used to determine the correct weight vectors to trace the rate regions calculated in the following section.

4.1.2 2 Users Near-Far ADSL Downstream

Once again, the scenario illustrated in Figure 2.6 be used to demonstrate the data rate performance of the Multi-User Greedy algorithm with rate bisection. The simulation parameters used are the same as those shown in Table 3.6.

The graph in Figure 4.2 shows the rate regions for this scenario achievable by OSB, ISB, IWF and finally, Greedy with weight bisection. The rate region performance of Greedy with bisection is seen to be quite close to the optimal solution given by OSB and slightly better than that of ISB. The performace of IWF is seen to be significantly poorer than the other three algorithms.

One particular data rate point is highlighted in table 4.1. In this example, the CO line rate is set at 600 bits per frame (2.4Mbps). At this rate point, ISB achieves a rate on the RT line within 5.7% of the optimal solution, IWF within

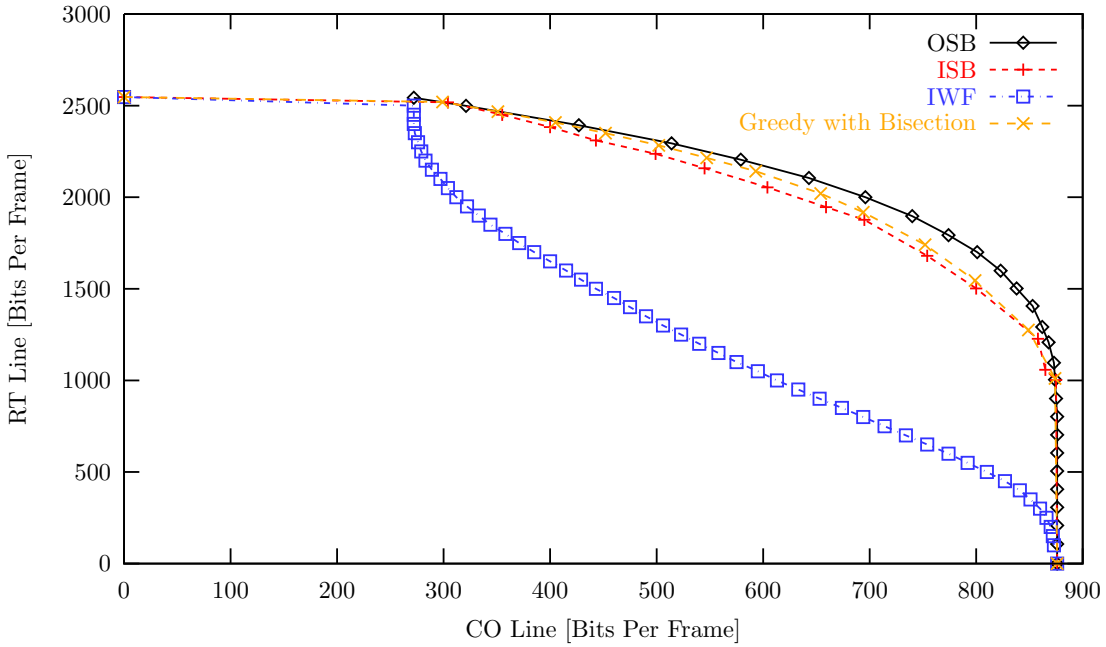


Figure 4.2: Comparison of rate regions for various loading algorithms compared to Greedy with bisection for the 2 User ADSL Downstream scenario in Figure 2.6

	OSB	ISB	IWF	Greedy
RT Line	2178	2054	1050	2142

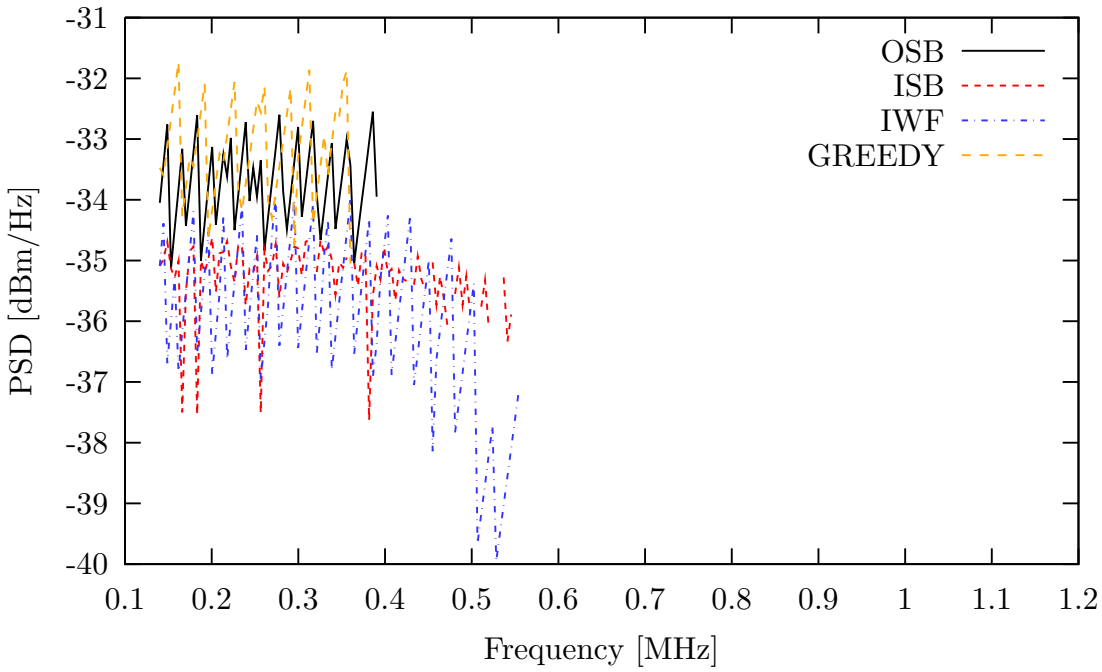
Table 4.1: Data rates for scenario in Figure 2.6 with the CO line rate fixed at 600 bits per frame

51.8% and greedy within just 1.7%.

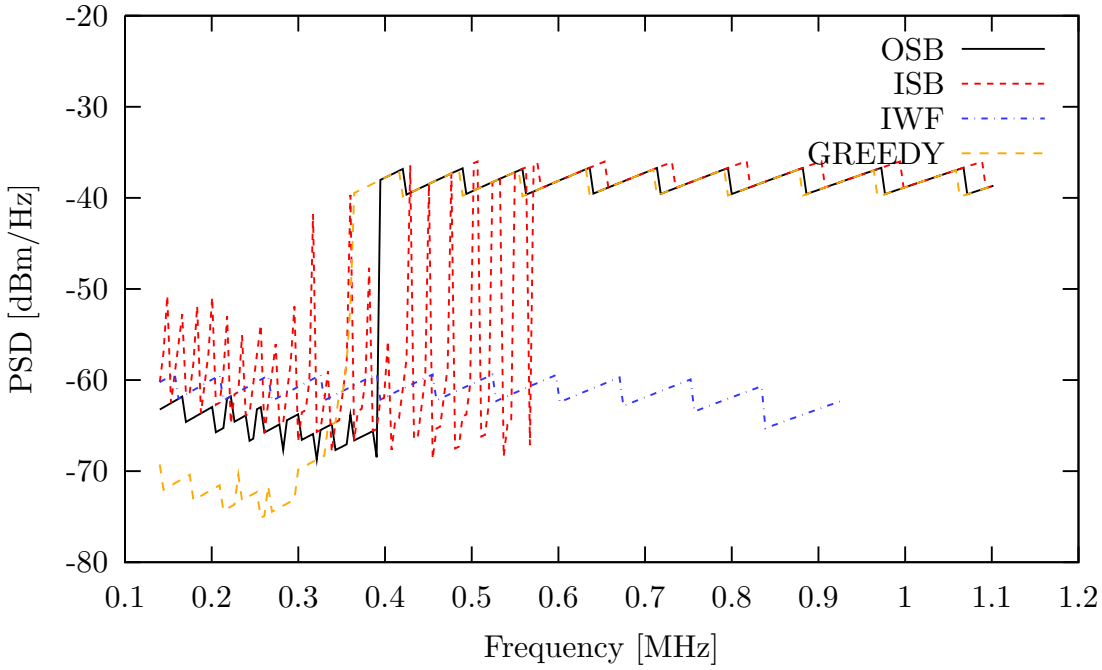
The spectra generated by each algorithm at this data rate point are shown on the same axis in Figure 4.3. On both the CO line and the RT line it can be seen that the greedy algorithm produces a spectrum which most resembles the optimal spectrum provided by OSB.

4.1.3 3 User VDSL 2 Upstream

The graph in Figure 4.4 shows the rate regions for the scenario in Figure 3.15 using the simulation parameters shown in Table 3.8. In these rate regions, the rates of user 1 and user 2 are traded against each other and the rate target for



(a) Line 1



(b) Line 2

Figure 4.3: Spectra generated for the scenario in Figure 2.6 by various algorithms with the CO Line data rate fixed at 600 bits per frame

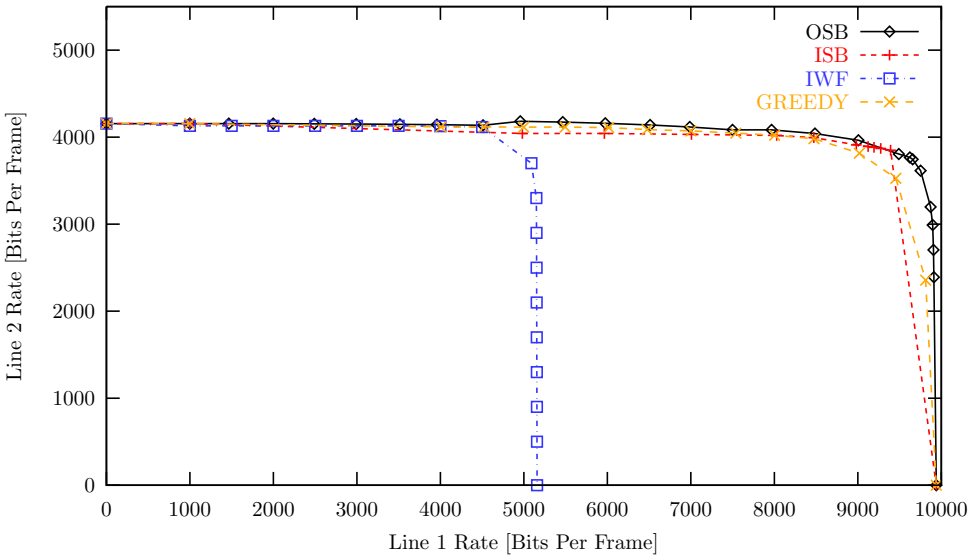


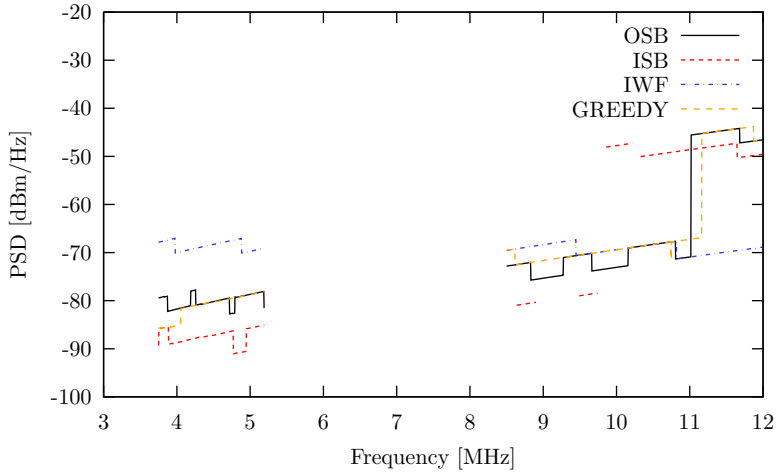
Figure 4.4: Comparison of rate regions for various loading algorithms compared to Greedy with bisection for the 3 User VDSL2 Upstream Scenario in Figure 3.15

the longest line is set close to its maximum date rate with no FEXT at 1400 bits per frame (5.6Mbps).

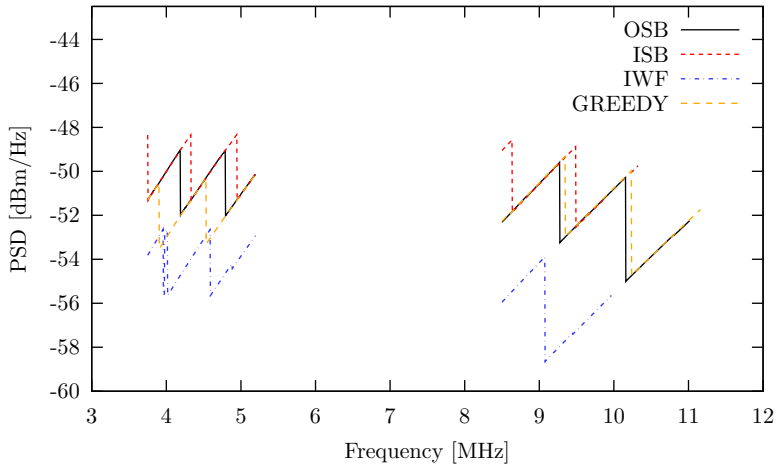
By inspecting the graph in Figure 4.4, the performance of the greedy algorithm is seen to be very close to that of OSB and ISB, whilst significantly outperforming IWF. The rate regions of OSB, ISB and the greedy algorithm are very similar in this case, tracing almost equally sized regions. IWF however, performs quite poorly. IWF can reach a maximum rate of 5150 bits per frame on line 1, almost 50% less than the maximum on line 1 for OSB.

The discontinuity in the ISB rate region at the end of its line 1 range is due to the fact that it is not possible to obtain a rate convergence for ISB in this region. It appears that there is a small region of non-convexity in w for ISB at this particular part of the rate region in this scenario.

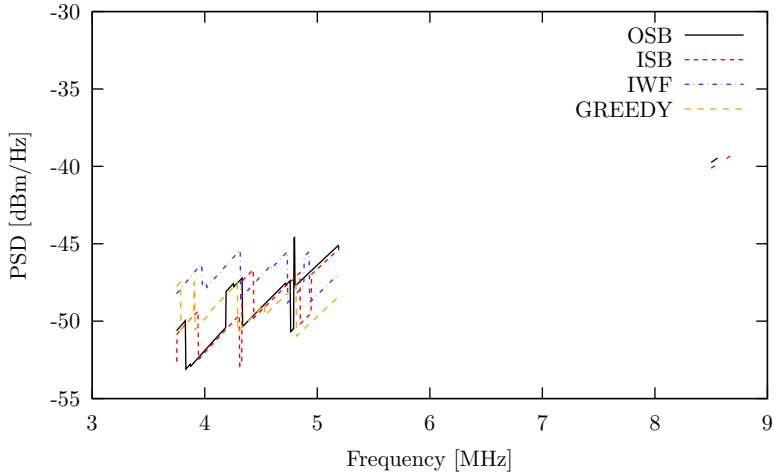
In Figure 4.5, the spectra for a particular data rate point on the rate region are shown. The data rates on lines 1 and 3 are fixed at 5000 and 1400 bits per frame respectively. The resulting data rates on line 2 are shown in Table 4.2. Once again, it is found that the greedy algorithm performs close to the optimal solution and is slightly better than ISB and significantly better than IWF. The spectrum on each line are illustrated in 4.5. It is clear that the spectrum generated by the



(a) Line 1



(b) Line 2



(c) Line 3

Figure 4.5: Spectra generated for the scenario in Figure 3.15 by various algorithms. The data rate on line 1 and line 3 are fixed at 5000 and 1400 bits per frame respectively

	OSB	ISB	IWF	Greedy
Line 2	4181	4042	3080	4124

Table 4.2: Data rates for scenario in Figure 3.15 with data rates of line 1 and 3 fixed at 5000 and 1400 bits per frame

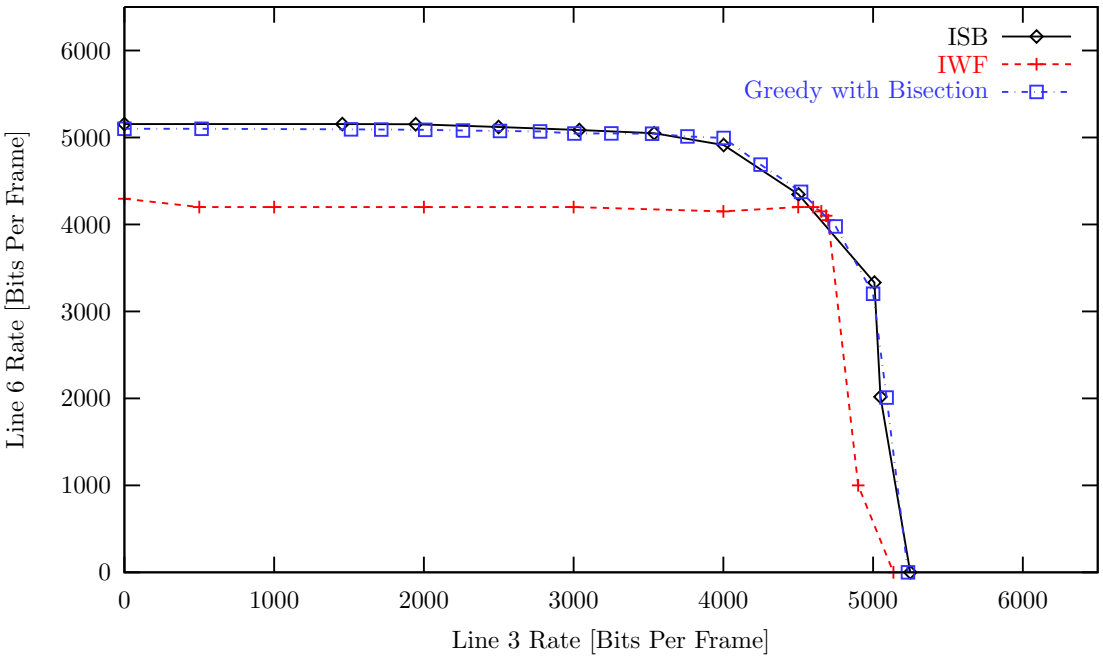


Figure 4.6: Comparison of rate regions for various loading algorithms compared to Greedy with bisection for the 3 User VDSL2 Upstream Scenario in Figure 3.18

greedy algorithm is very close to the optimal solution in this case and much closer than that of ISB or IWF.

4.1.4 6 User ADSL 2+ Downstream

The graph in Figure 4.6 shows the rate regions for the scenario in Figure 3.18 using the same simulation parameters as shown in Table 3.10. In the rate regions shown, the rate combinations of user 3 and user 6 are shown while the rate targets of the other lines are fixed at 2600 bits per frame (10.4Mbps), 2000 bits per frame (8 Mbps), 5300 bits per frame (21.2 Mbps) and 7100 bits per frame (28.4 Mbps) respectively.

In Figure 4.6, the greedy algorithm is shown alongside IWF and ISB. Due to the high complexity of OSB, it would have been prohibitively time consuming to obtain a rate region for OSB for this scenario so it is not included.

The greedy algorithm is shown to have very similar performace to ISB and significantly better performance that IWF. By deduction from the previous rate regions, it is reasonable to assume that once again the greedy rate region is comparable to that of the optimal solution.

Figure 4.7 shows the spectra produced by ISB, IWF and the greedy algorithm for this scenario. It is observed that the PSDs of ISB and greedy are very similar. IWF produces similar spectra on lines 1,2 and 4 but shows a marked difference on lines 3, 5 and 6. These differences in the PSDs account for the poor performance of IWF at this data rate point.

4.2 Sub-Gradient Weight Search

An alternative rate searching algorithm to the bisection method is a sub gradient search. Stated simply, the sub-gradient search adjusts the current value of the weight by some fraction of the current distance from the rate target.

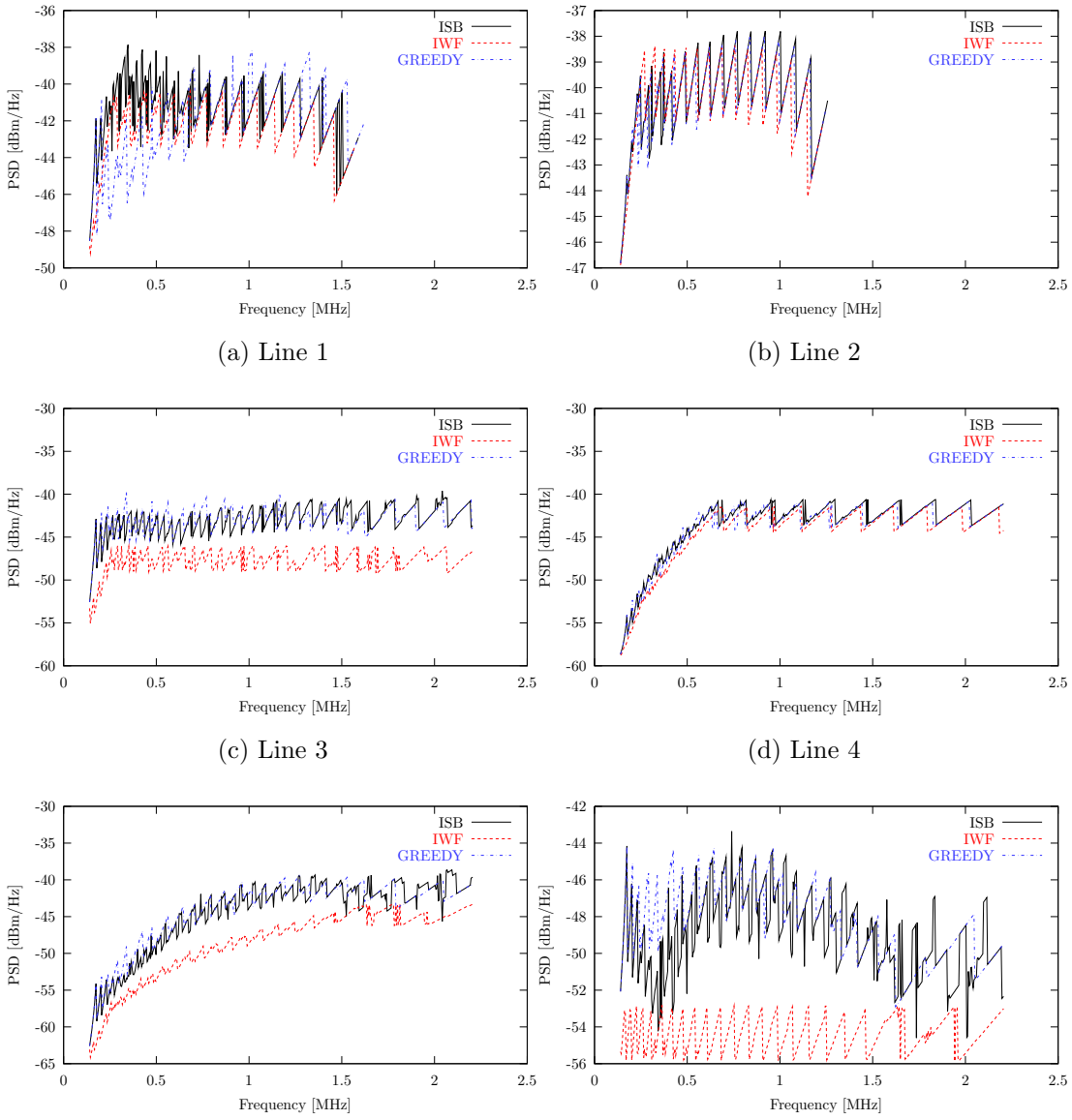
$$w_n^{+1} = w_n + \epsilon(R_n - R_n^{target}) \tag{4.1}$$

The main advantage of a sub-gradient method over bisection is that the weights of all users can be updated simultaneously, potentially requiring a lower number of iterations to converge.

Algorithm 16 Sub-Gradient Weight Search for Greedy Algorithm

$w_n = 1, \forall n$
repeat
    Execute Greedy Algorithm
    $w_n^{+1} = w_n + \epsilon(R_n - R_n^{target}), \forall n$
until All Rate Targets Met

The disadvantage is that the correct step is generally not known a-priori. A very high step size ϵ can result in fast convergence but may cause oscillations



(a) Line 1

(b) Line 2

(c) Line 3

(d) Line 4

(e) Line 5

(f) Line 6

Figure 4.7: PSDs generated by ISB, IWF and Greedy for the 6-User ADSL2+ scenario in Figure 3.18

	4 User		8 User
Line 2	1900 (7.6 Mbps)	Line 2	800 (3.2 Mbps)
Line 3	4700 (18.8 Mbps)	Line 3	2500 (10 Mbps)
Line 4	4300 (17.2 Mbps)	Line 4	3000 (12 Mbps)
		Line 5	4000 (16 Mbps)
		Line 6	3800 (15.2 Mbps)
		Line 7	900 (3.6 Mbps)
		Line 8	1800 (7.2 Mbps)

Table 4.3: Rate targets for 4 and 8 User ADSL2+ scenarios used for results in Figure 4.8

around the point of convergence, effectively rendering the algorithm useless. A very low step can guarantee convergence, but may result in a very slow convergence time. The sub-gradient search algorithm used along with the greedy bit-loading algorithm is shown in Algorithm 16.

In Figure 4.8, a comparison between bisection and sub gradient search for the greedy algorithm is illustrated for two DSL scenarios. The first is a four user ADSL2+ downstream scenario which uses the first four lines of the 6-User ADSL2+ scenario shown in Figure 3.18. The second is an eight user ADSL2+ downstream scenario which uses the same configuration as Figure 3.18 with the first two lines repeated. The rate targets for each line in the two scenarios are shown in Table 4.3. In both cases, the rate of Line 1 is left to be maximised. Illustrated in Figure 4.8 is the number of executions of the greedy algorithm required to complete the rate search for both scenarios against the step size ϵ. Two straight lines are also shown in Figure 4.8 representing the number of executions required by the bisection algorithm for each scenario from Algorithm 15. The decrease in convergence time as the step size increases is evident for both scenarios. However, for both scenarios the final step size values shown are the limit of stability for the sub gradient search above which the rate search does not converge.

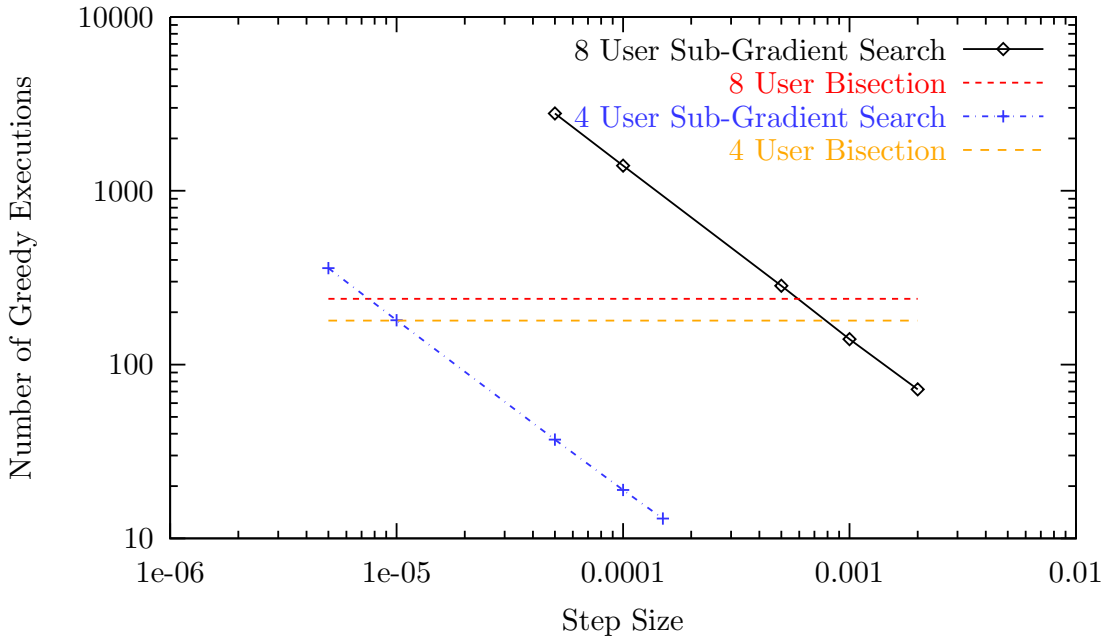


Figure 4.8: Number of executions of the greedy algorithm required for two scenarios against the step size ϵ. Also shown is the number required using the bisection method for each scenario

It also clear from Figure 4.8 that there is no universally good value for the step size, so it may be a risk to choose sub gradient search over bisection. problem will be presented.

4.2.1 Adaptive Sub-Gradient Search

In order to combat the problem of step size selection in a sub-gradient rate search, in this section, a new algorithm is presented. This algorithm utilises a sub-gradient search with an adaptive step size. Simulation results are presented and a significant performance advantage over bisection is observed without stability problems.

Figure 4.9 illustrates a sub-gradient rate search for two different 3-User scenarios. In this case, there are rate targets set for line 1 and 2 and the rate on the third line is subsequently maximised. Shown in Figure 4.9 are both a well behaved sub-gradient search and a badly behaved one, which exhibits oscillatory behaviour. The boxes represent the range in which the rate targets are within

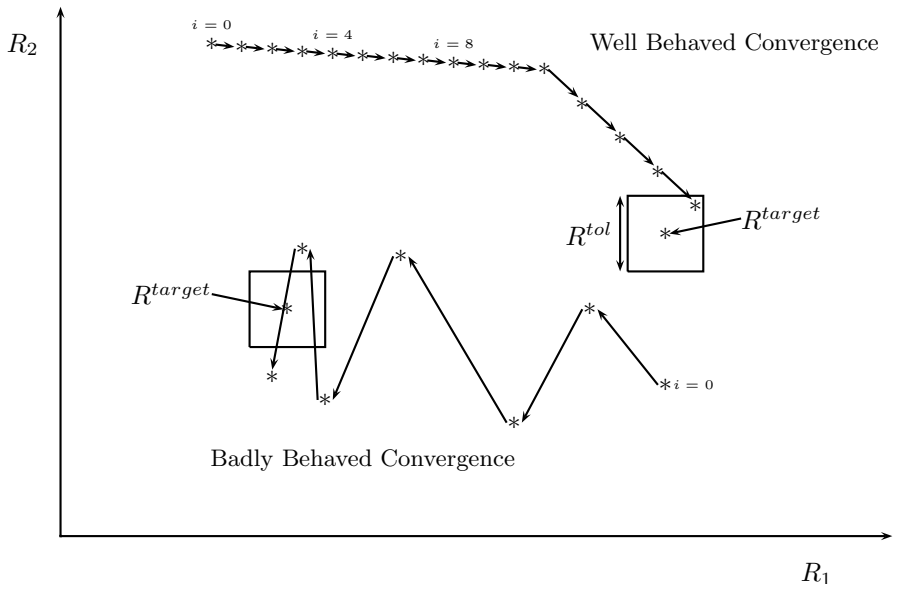


Figure 4.9: Illustration of a sub-gradient search of the weight vector w

an acceptable tolerance. With the standard sub-gradient search algorithm from Algorithm 16 it is not possible to know a-priori whether the value of ϵ will lead to slow well behaved convergence, fast convergence or not converge due to oscillations.

A real example of sub-gradient rate search iterations is illustrated in Figure 4.10. Since this can only be visualised in two dimensions (see Section 2.9.1), the scenario chosen is the 3-User Upstream VDSL scenario from Figure 3.15. The rate targets are set for lines 1 and 3 as 8000 and 1400 bits per frame respectively. The rate of the line 2 will be maximised. The rectangle in Figure 4.10 represents a tolerance of 80 bits per frame on each line.

Two values of ϵ are utilised in this scenario to demonstrate well behaved convergence and oscillation. Both traces in Figure 4.10 start at the bottom rightmost point on the graph before moving towards the rate target. The trace which utilises a value of $\epsilon = 1e - 5$ is seen to exhibit slow but direct convergence to the rate targets. The trace which utilises a value of $\epsilon = 0.0085$ is seen to oscillate around the rate targets and in fact does not converge.

An algorithm which attempts to overcome the step size selection problem for

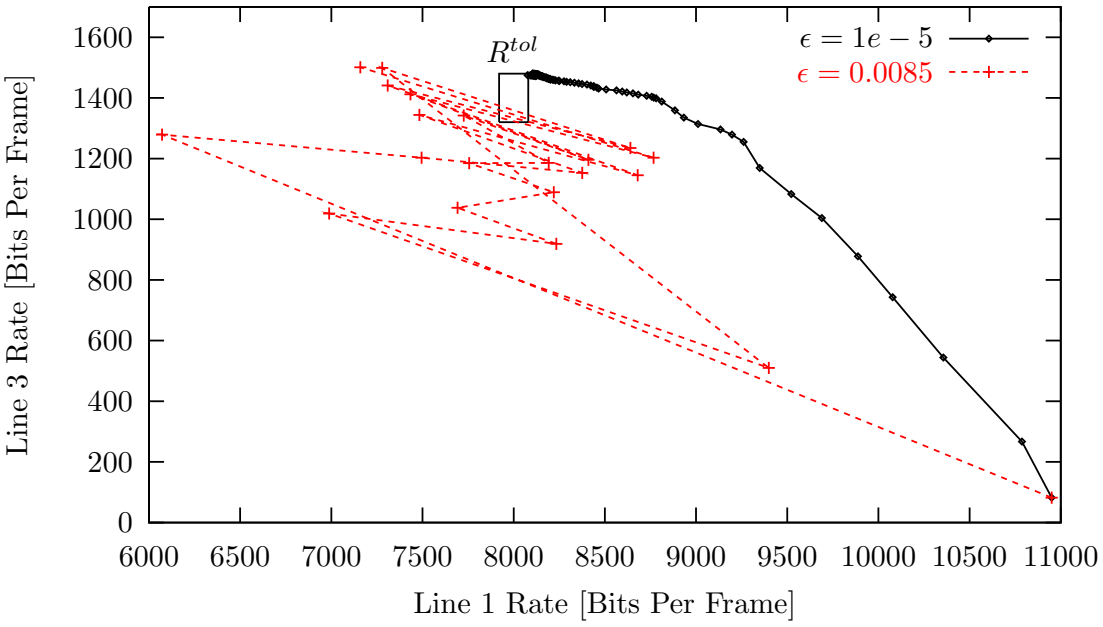


Figure 4.10: Illustration of iterations of sub gradient rate search for 3-User Upstream VDSL scenario in Figure 3.15

sub-gradient search is shown in Algorithm 17. The algorithm works by starting from a very low value of ϵ and increasing it until oscillatory behaviour is detected. Oscillatory behaviour is defined as two lines moving from one side of their rate target to the other during one iteration of the algorithm. Once oscillatory behaviour is detected, the algorithm moves back one iteration to the previous rate point and starts again with a smaller step size.

Figure 4.11 illustrates the operation of Algorithm 17 for the 3-user upstream VDSL scenario in Figure 3.15. The rate targets and tolerances are identical as those used in the sub-gradient searches in Figure 4.10. It is clear from this trace that the adaptive sub-gradient algorithm reaches the rate targets in a much more efficient manner than that of the small ϵ sub-gradient search in Figure 4.10, without exhibiting the oscillatory behaviour given by a large value of ϵ.

Figure 4.12 shows the number of greedy executions required by the bisection algorithm and by the adaptive sub-gradient method against the number of users for a downstream ADSL2+ scenario. For this experiment, the line parameters used are repeating patterns of the 6-User scenario in Figure 3.18. The rate targets

Algorithm 17 Adaptive Sub-Gradient Weight Search for Greedy Algorithm

$w_n = 1, \forall n$
$\epsilon = 1e-7$
Execute Greedy Algorithm
$w_n^{+1} = w_n + \epsilon(R_n - R_n^{target}), \forall n$
repeat
  $osc = 0$
  Execute Greedy Algorithm
  for $n = 1 \dots N$ **do**
    if $R_n > R_n^{target} \cap R_n^{-1} < R_n^{target}$ **then**
      $osc = osc + 1$
    else if $R_n < R_n^{target} \cap R_n^{-1} > R_n^{target}$ **then**
      $osc = osc + 1$
    end if
  end for
  if $osc \geq 2$ **then** ▷ Two or more lines rates are oscillating
    $\epsilon = \epsilon \div 2$ ▷ Reduce step size by half
    $w_n^{+1} = w_n^{-1} + \epsilon(R_n^{-1} - R_n^{target}), \forall n$
    ▷ Use previous value of w and R with new ϵ
  else
    $\epsilon = \epsilon \times 2$ ▷ Increase step size by two
    $w_n^{+1} = w_n + \epsilon(R_n - R_n^{target}), \forall n$
  end if
until All Rate Targets Met

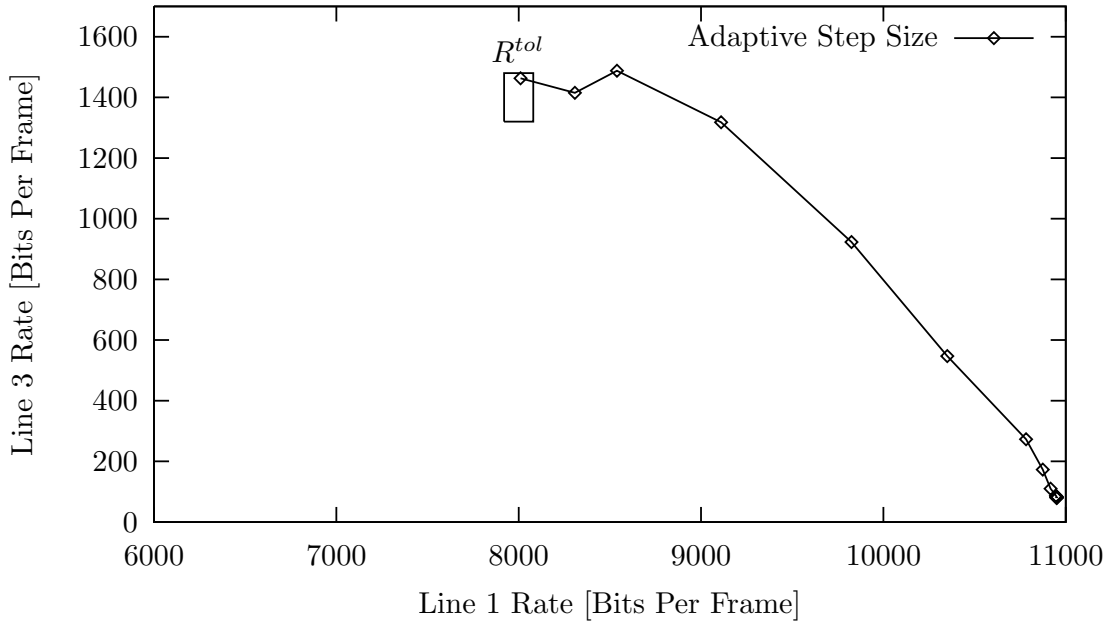


Figure 4.11: Illustration of iterations of the adaptive sub-gradient rate search for 3-User Upstream VDSL scenario in Figure 3.15

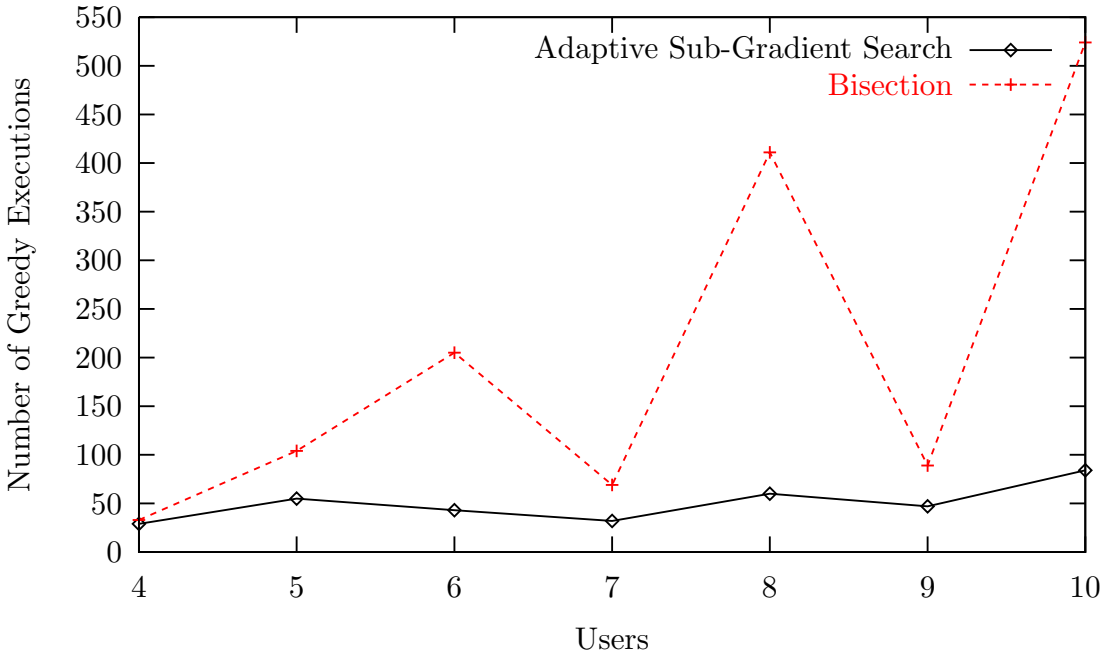


Figure 4.12: Number of greedy executions required for a bisection rate search and the adaptive sub-gradient rate search

used are shown in Appendix A.1. It is clear that the adaptive sub-gradient algorithm significantly outperforms the bisection method for all the scenarios tested. It also appears to scale more gracefully than bisection with the number of users, meaning that the speedups will be larger for increasing number of users. The peaks and troughs in figure 4.12 are caused by the particular rate targets chosen for each scenario which take varying numbers of iterations to converge. Different sets of rate targets would result in the peaks and troughs being distributed differently.

4.3 PSD Vector Caching

Disregarding the initial cost matrix calculation, the greedy algorithm requires approximately

$$B(O(N^4) + O(K))) \tag{4.2}$$

operations, where B is the total number of bits loaded. The $O(N^4)$ term is the cost of calculating the new PSD values on a particular sub-channel. This is an $O(N^3)$ operation, executed N times.

The complexity expression in expression 4.2 above is only true for one iteration of the greedy algorithm. When a rate target search is carried out by either bisection or sub-gradient search, the greedy algorithm will be executed many times in succession. The complexity in this case will be approximately $T^w B\dot{(}O(N^4) + O(K)))$ where T^w is the number of iterations required in the rate search.

Considering the fact that the values of the channel gains and crosstalk gains are identical between iterations of the greedy algorithm (i.e. the system does not change between executions), it is reasonable to assume that some of the same PSD vectors will calculated for a particular tone in separate iterations of the rate search. In other words, the algorithm will need to calculate the PSD vector for a particular bit vector that has been calculated in a previous iteration of the algorithm.

Given that the PSD vector calculation is a $O(N^3)$ operation, it is posited that there might be a significant speed improvement possible if the algorithm were to cache the values of the PSD vectors that is has already calculated.

Figure 4.13 shows an illustration of the algorithm operation visualised as a flow diagram. The update w values step in the diagram can be a per user bisection or a sub-gradient search. As shown in the diagram, the update Δp step checks if the current PSD to be calculated has already been calculated and therefore present in the cache. If the result has not been cached it is added to the cache.

Figure 4.15 shows the execution times obtained using the greedy algorithm with a bisection rate search against number of users with and without PSD

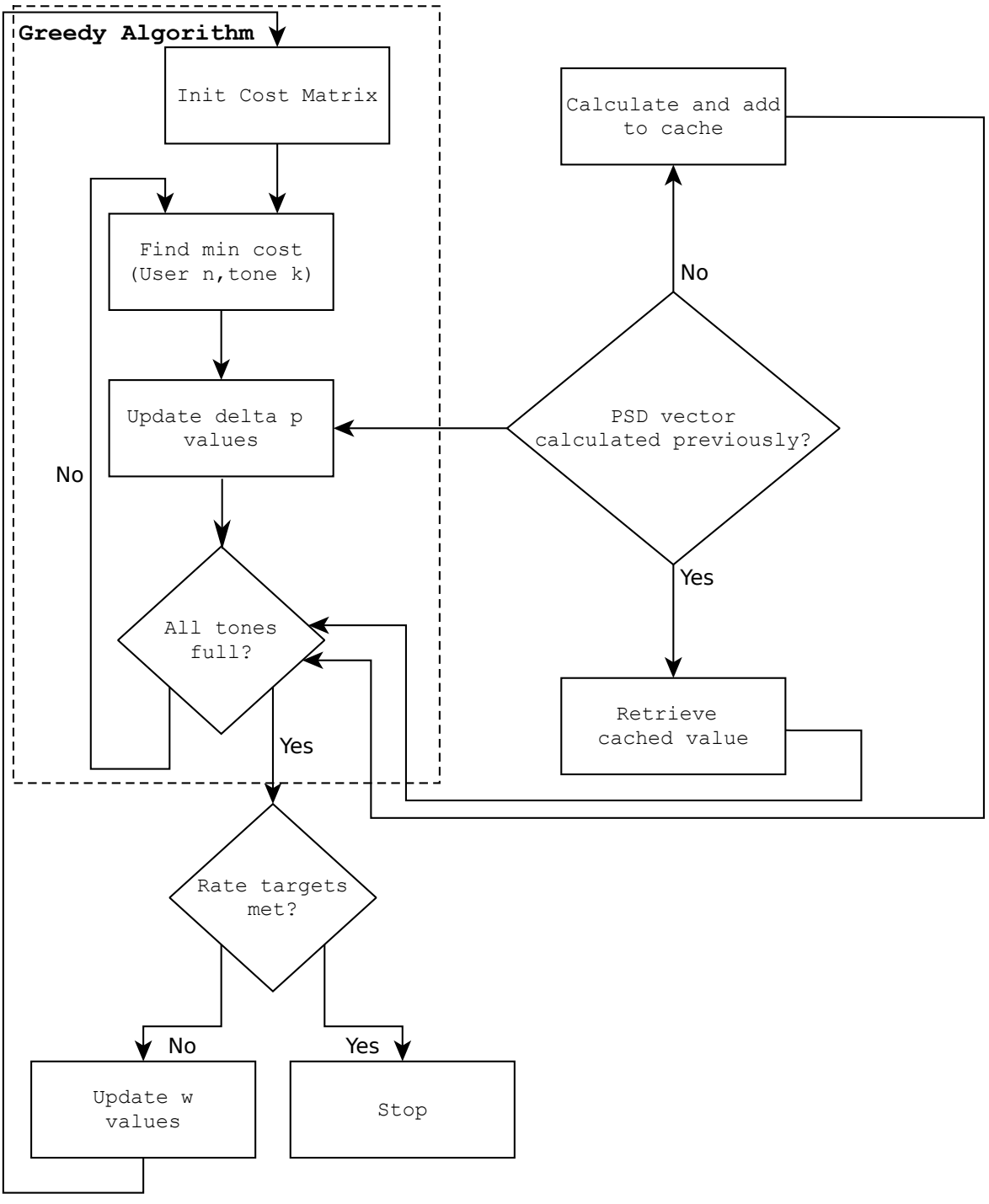


Figure 4.13: Flow chart showing operation of the greedy algorithm with rate search and PSD caching

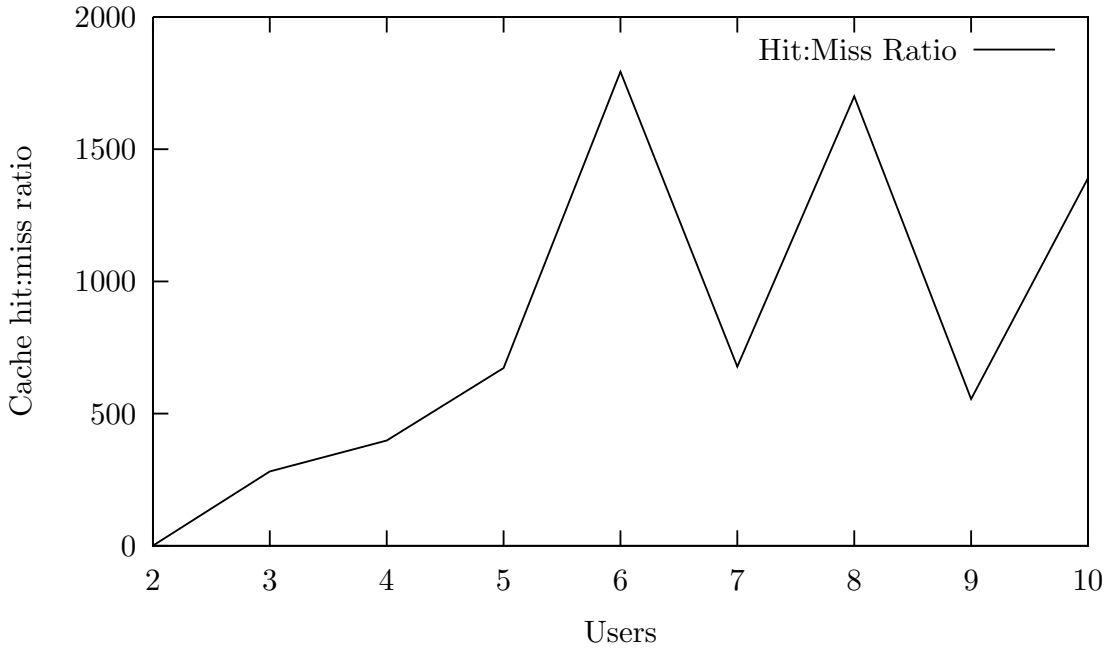


Figure 4.14: Cache hit:miss+collision ratio for the PSD Vector cache against number of ADSL users obtained from profiling the cache during simulation

caching. The rate targets for each set of users in shown in Appendix A.1. For ease of implementation, a fixed size cache with a maximum total of 38.4 million entries was used, that is 80000 entries per tone on 480 tones (ADSL 2+ downstream). The cache utilises a hash table on each tone with a fixed number of buckets (hash table entries) set at 80000. This reduces the complexity of the implementation but will inevitably lead to some collisions in the hash table. The cache uses a Most Recently Used (MRU) replacement strategy, where the most recent value is entered into the cache if there is a collision. This is chosen because it is expected that small changes in the weights during the rate search will lead to similar behaviour of the greedy algorithm, resulting in a fewer total number of collisions in the cache.

Figure 4.15 shows that a significant speedup is obtained when using PSD Vector caching. Figure 4.16 shows the absolute speedup obtained against the number of users, which is as large as 14 times faster for 8 users. The number of greedy executions required in each case is also illustrated. It is expected that a larger speedup would be observed when a larger number of greedy executions

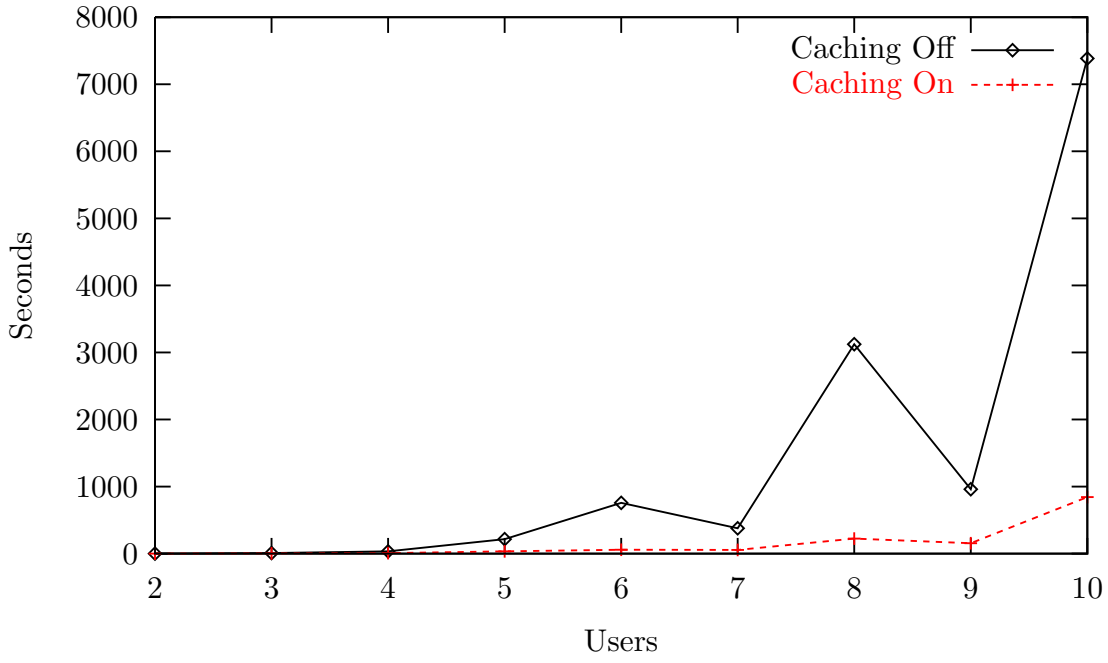


Figure 4.15: Execution time for greedy algorithm with bisection rate search against number of users with caching on and off

occur, resulting in a larger number of possible cache hits and in fact this behaviour is present in Figure 4.16.

In Figure 4.16 it is found that the correlation between number of executions and the absolute speedup begins to drop off at 10 ADSL users. This behaviour is observed because of the limitations of the cache size utilised in these experiments. By observing Figure 4.14 a drop in the cache hit:miss+collision ratio can be seen at 10 users. This is due to using a fixed bucket size (number of hash table entries) which starts to see a noticeable amount of cache collisions at 10 users. Using a large enough cache would mitigate this issue.

Figure 4.17 shows the total number of entries in the cache against the number of users. A best fit lines of aN^3 is also shown in Figure 4.17. It is believed that the number of entries in the cache should grow approximately proportional to N^3 as the number of bits loaded grows approximately in N and the number of bisection steps in N^2. If we assume that the best fit line represents the growth in cache size, then it follows that for 50 users the cache will contain approximately 10^9 entries.

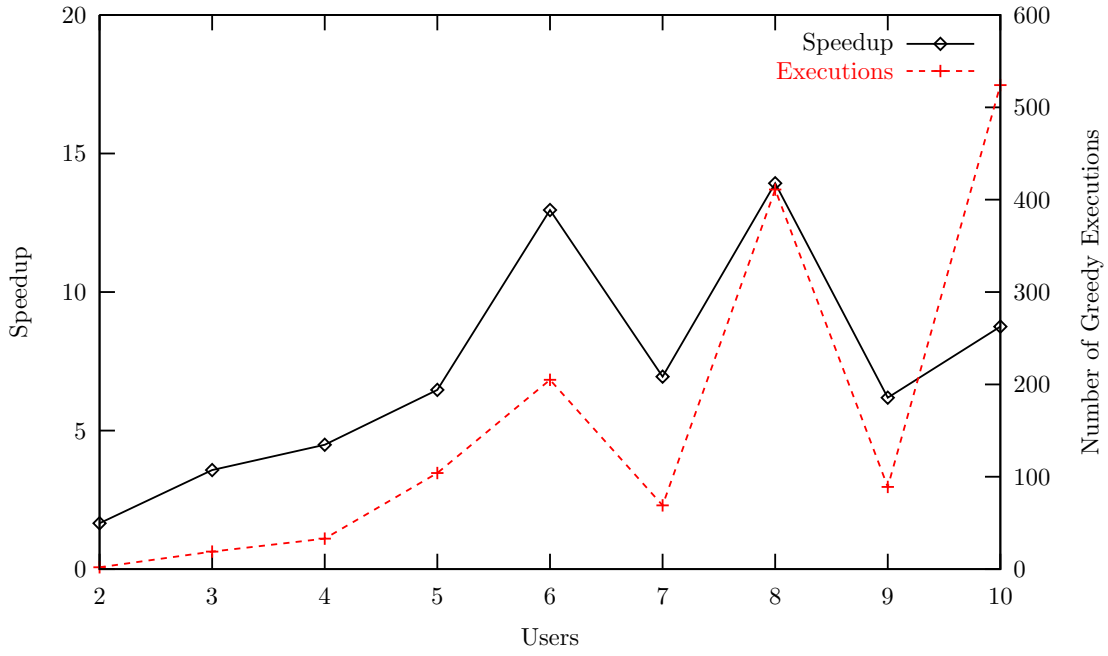


Figure 4.16: Speedup of PSD Vector caching over non PSD Vector caching greedy algorithm against number of users. Also shown is the number of executions of the greedy algorithm required

Hash	4 bytes
Bit Vector	50 * 4 bits = 25 bytes
PSD Vector	50 * 16 bits = 100 bytes

Table 4.4: Data structures present in the cache entry

Since the maximum power on any tone is $-36.5dBm/Hz(1mW)$, it is assumed that each power value can be adequately described in 10000 quantisation levels (much higher than the resolution required by the standard of 0.5dBm/Hz [47]) which equates to about 13.3 bits. Rounding up slightly, 2 bytes is used for each power value. Each bit value on each tone is in the range 0-15 and can therefore be stored in 4 bits. Using these values, the size of a cache entry for a 50 user network is approximately 129 bits as illustrated in Table 4.4.

Combining this value with the predicted number of cache entries gives an approximate figure of 130Gbytes for an ADSL2+ network.

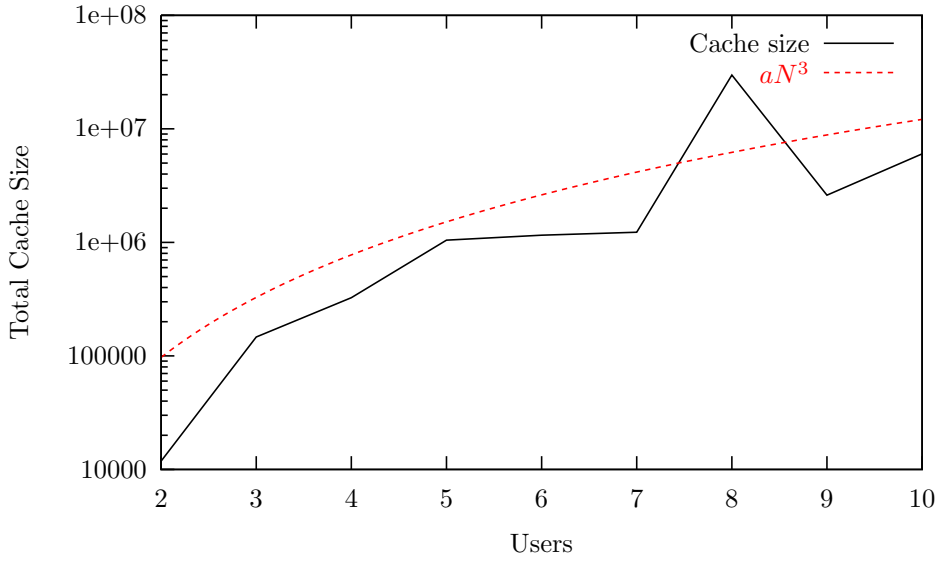


Figure 4.17: PSD Vector cache size against number of users for the greedy algorithm with rate bisection

4.4 Parallelisation

In this section, a new algorithm will be presented that enables parallelisation of the multiuser greedy algorithm rate search. By designing the algorithm to exploit the parallelism offered by modern processors, it is possible that the speed of the algorithm can be improved.

4.4.1 Parallel M-Section

This algorithm is a parallel version of the bisection algorithm which is hereafter known as parallel m-section. The basic idea is to divide the region of interest in the w search into smaller areas which are then evaluated in parallel. This reduces the convergence time for the per-user w search.

Figure 4.18 shows an illustration of a traditional bisection algorithm. The values of $w^0 \cdots w^3$ in Figure 4.18 represent the iterations of a bisection algorithm. Starting from an arbitrary initial value, the value w is reduced by a factor of 2 until a value of w which falls on the other side of the target rate is found. This value is now w^{min}. From this stage, the algorithm proceeds to traditional bisection and is shown to find a value with the required rate tolerance in one more iteration.

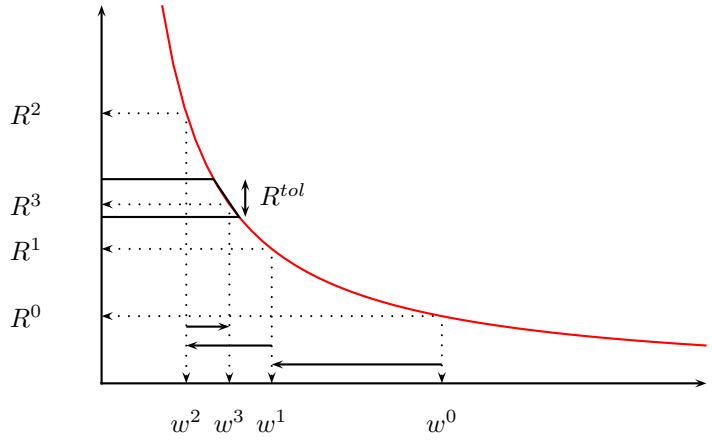


Figure 4.18: Illustration of a traditional bisection of the weight vector w

The illustration in Figure 4.18 is described algorithmically in Algorithm 15.

An illustration of two iterations of a parallel m-section with two threads is shown in Figure 4.19. At each iteration, the values of w tested are spaced equally between the values of w^{max} and w^{min}. It is assumed that values for w^{max} and w^{min} are already known, but this may not be the case in practise.

Having completed the first iteration in Figure 4.19 new values for w^{max} and w^{min} are found. This region is then divided up equally between the available threads and the process repeated until a value of w is found which meets the rate target on that line.

After each completed bisection of w, at least one of w^{min} or w^{max} is known for the next rate bisection by examining the results from the previous iteration. The only case where this is not true is on the very first bisection before the first execution of the greedy algorithm and in this case the initial values of w^{max} and w^{min} have to be selected. In the ordinary weight bisection algorithm the first value of w is always 1. In the parallel version with M threads, a method is required for selecting the M initial values of w.

In order to achieve this two functions are chosen to represent the initial values of w^{max} and w^{min}. It was found experimentally that in order to achieve better performance with increasing an increasing number of threads, the gaps between the initial w should decrease as the number of threads increases. This is achieved

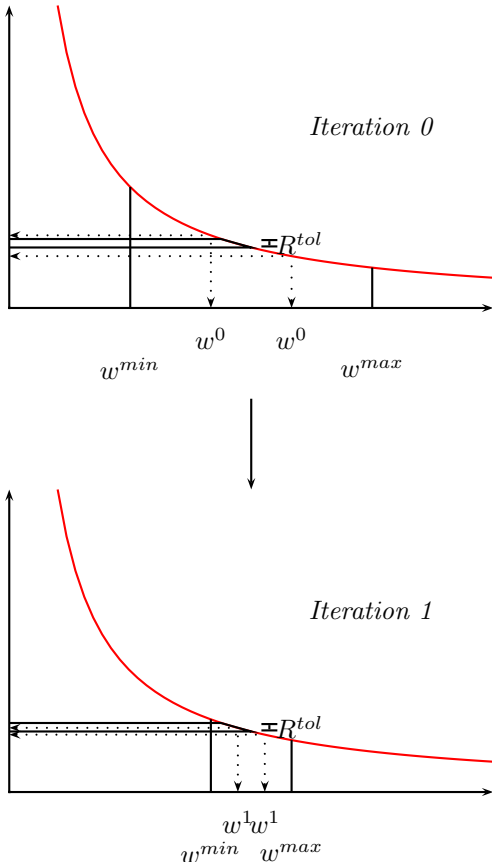


Figure 4.19: Illustration of a parallel m-section of the weight vector w with two threads

by choosing a lower function of:

$$w^{min} = \frac{1}{2^{\frac{M}{4}}} \tag{4.3}$$

and an upper function of:

$$w^{max} = \sqrt{M} \tag{4.4}$$

These functions are illustrated in Figure 4.20. Shown in Figure 4.20 are the initial w values chosen for each number of threads based on these upper and lower values.

The parallel m-section algorithm is shown in Algorithm 18 and illustrated in flow chart form in Figure 4.21.

Performance results for the parallel m-section algorithm are shown in figures 4.22 and 4.23. A significant performance increase is observed which flattens out

Algorithm 18 Parallel M-section Rate Search for Greedy Bit Loading

repeat
  for $n = 1 \dots N$ **do**
    if Iterations=0 **then**
      $w_{bottom} = \frac{1}{1^{\frac{M}{4}}}$
      $w_{top} = \sqrt{M}$
    else
      if $R_n^{last} > R_n^{target}$ **then**
        $w_{bottom} = w_n^{last}$
        $w_{top} = w_{bottom} * 2$
      else
        $w_{top} = w_n^{last}$
        $w_{bottom} = w_{top}/2$
      end if
    end if

    repeat
      if Iterations=0 **then**
        for $m = 1 \dots M$ **do**
          $w_n = w_{bottom} + \frac{w_{top} - w_{bottom}}{M+1} \times m$ ▷ See Note 1
          Execute Greedy Algorithm m ▷ In parallel
        end for
      else
        for $m = 1 \dots M$ **do**
          $w_n = w_{bottom} + \frac{w_{top} - w_{bottom}}{M-1} \times (m-1)$ ▷ See Note 2
          Execute Greedy Algorithm m ▷ In parallel
        end for
      end if
      $w_{bottom} = \text{argmin}_{w_n} R_n^m - R_n^{target}$
      $w_{top} = \text{argmin}_{w_n} R_n^{target} - R_n^m$
    until $|R_n - R_n^{target}| < R^{tol}$
  end for
until Rate Targets Met

Note 1: On first iteration use the starting values of w evenly spaced with $M+1$ gaps between w_{top} and w_{bottom}
Note 2: On subsequent iterations use starting values evenly spaced with M gaps between w_{top} and w_{bottom}

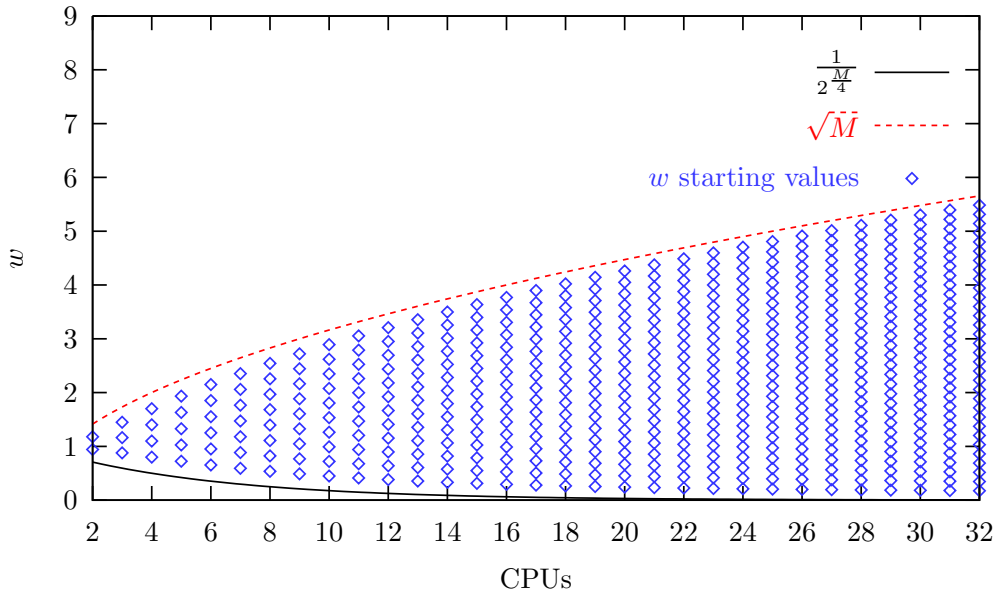


Figure 4.20: Functions for w_{max} and w_{min} plotted against total number of CPUs available

as the number of threads increases. The magnitude of the speedup also appears to increase with the number of users so we would expect to see an even larger speedup for most practical DSL scenarios.

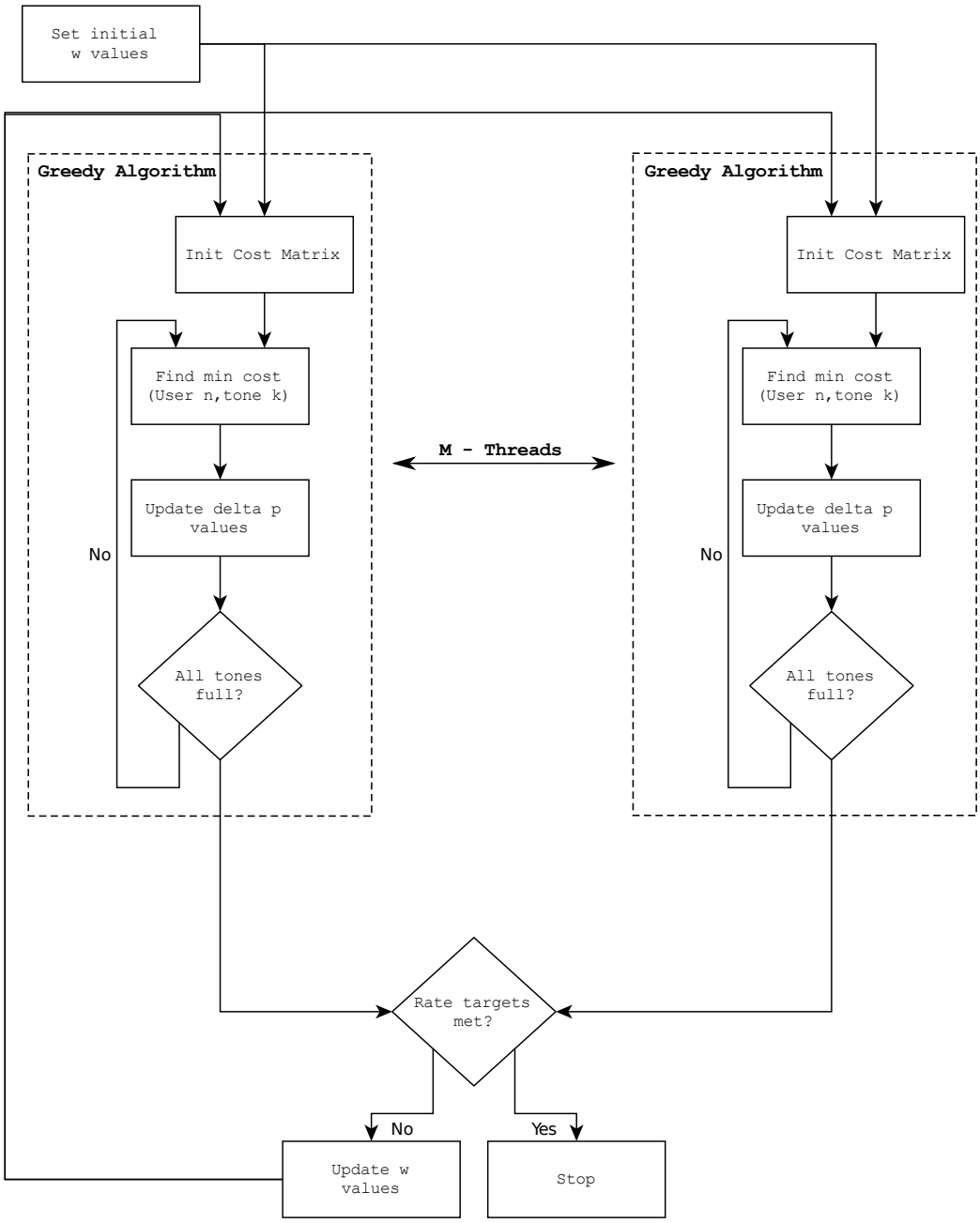


Figure 4.21: Flow diagram of parallel m-section algorithm

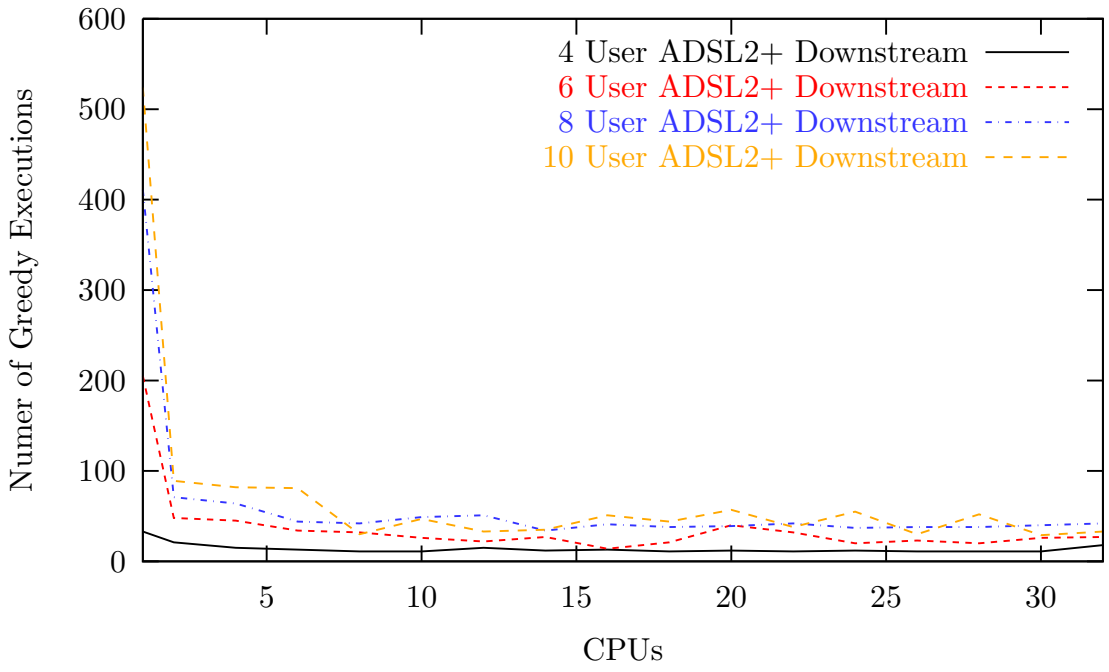


Figure 4.22: Number of greedy executions against number of CPUs for parallel M-section algorithm

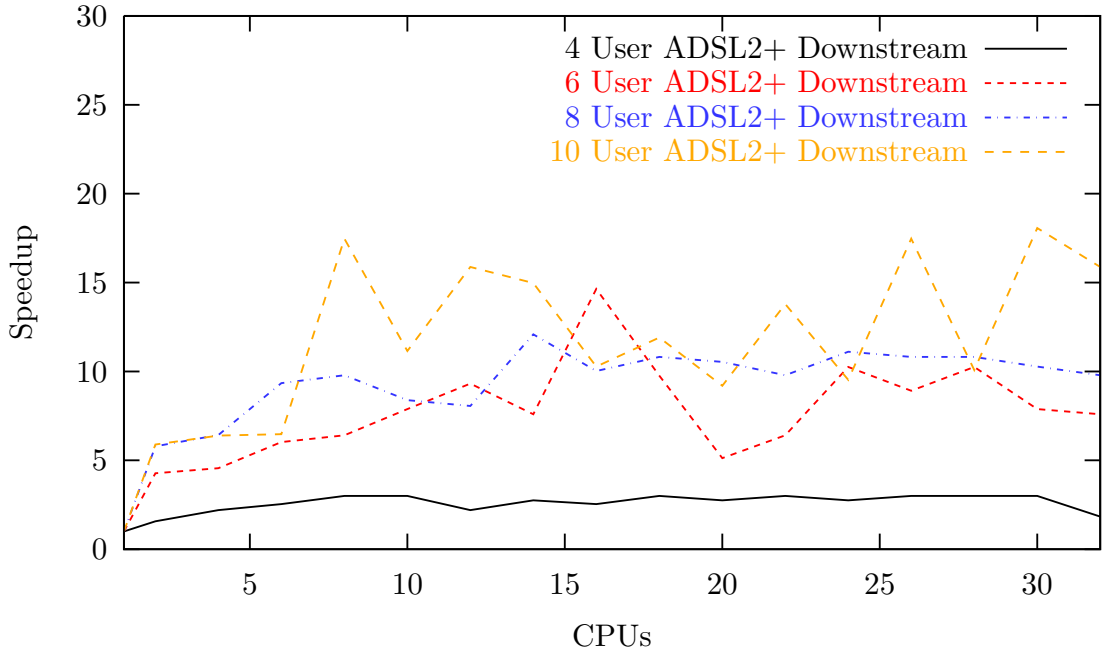


Figure 4.23: Speedup against number of CPUs for parallel M-section algorithm

4.5 Conclusions

In this chapter a number of enhancements to the Multiuser greedy algorithm were presented. By combining these techniques and high specification hardware, it is feasible that the greedy algorithm may be used for centralised DSM in the near future.

Using a bisection rate search, it was shown that the rate region performance of the greedy algorithm is close to the optimum and better than IWF and ISB for a range of DSL scenarios.

It was also shown that a sub-gradient rate search is possible and an adaptive sub-gradient rate search algorithm was presented to overcome the problem of convergence and step size selection. The adaptive sub-gradient was shown to be stable and exhibit a large performance advantage over bisection.

It was shown that a PSD cache could improve the performance of the greedy algorithm significantly. It is posited that a PSD cache for use in practical DSL scenarios (50+ lines), although very large, will be possible with high end hardware relatively soon. The PSD caching technique is also valid for other centralised DSM algorithms.

A parallel rate search algorithm was presented that can take advantage of multiple CPUs. This algorithm was shown to give large performance advantages over a single threaded bisection, but it probably not as efficient as using an adaptive sub-gradient approach.

Table 4.5 illustrates the execution times of various DSM algorithms which were obtained from simulation times on a Dell Inspiron 530n with a 2.33GHz Intel Core 2 E6550 with 2Gb of RAM. For the results including PSD caching, the cache parameters used were identical to those used for the results in Section 4.3. The DSL network configuration is taken from Figure 3.18 and the pattern is repeated for extra lines as necessary. The rate targets used are listed in Appendix A.1.

The data from Table 4.5 is visualised in Figure 4.24. The intractability of OSB, ISB and BBOSB are easily seen, which require over 1×10^5 seconds and

above for anything over 4 users. From this figure, the huge performance gains obtained using the various Greedy Algorithms are very apparent. For example, when using the Greedy Algorithm with an Adaptive Sub-Gradient rate search and PSD vector caching for a 7 User ADSL2+ scenario, a speedup of 3850 is obtained. The execution times in seconds in Table 4.5 do have not any particular significance individually but rather are used to compare the algorithm complexity on a reference platform.

Future work should seek to exploit parallelism in other parts of the algorithm to further decrease the execution time and make centralised DSM a realistic possibility.

Users	OSB	OSB + PSD Caching	ISB	ISB + PSD Caching	BBOSB	BBOSB + PSD Caching
2	35.65 s	14.6 s	32.21 s	2.73 s	7.12 s	2.88 s
3	3.24 hrs	2.12 hrs	16.22 mins	53.57 s	15.73 mins	4.2 minutes
4	†	*	3.93 hrs	9.35 mins	13.04 hrs	3.12 hours
5	*	*	12.06 hrs	30.04 mins	> 1 week	*
6	*	*	8.69 hrs	26.74 mins	*	*
7	*	*	28.46 hrs	1.41 hrs	*	*
8	*	*	*	†	*	*

* - not completed due to very long running time

† - could not be accurately calculated due to the large cache sizes causing swapping to disk on the simulation machine

Users	Greedy Bisection	Greedy Bisection + PSD Caching	Greedy Adaptive Sub Gradient + PSD Caching
2	0.26 s	0.199 s	0.78 s
3	9.99 s	3.89 s	5.46 s
4	35.6 s	10.96 s	10.29 s
5	3.71 mins	49.97 s	30.67 s
6	12.88 mins	1.7 mins	22.67 s
7	6.36 mins	1.21 mins	26.65 s
8	51.72 mins	5.67 mins	1.86 mins
9	15.92 mins	3.13 mins	1.41 mins
10	2.05 hours	13.14 mins	3.30 mins

Table 4.5: Execution times for various level 2 DSM algorithms using rate targets from Appendix A.1 and network configuration from Figure 3.18 repeated and the 1% worst case FEXT model

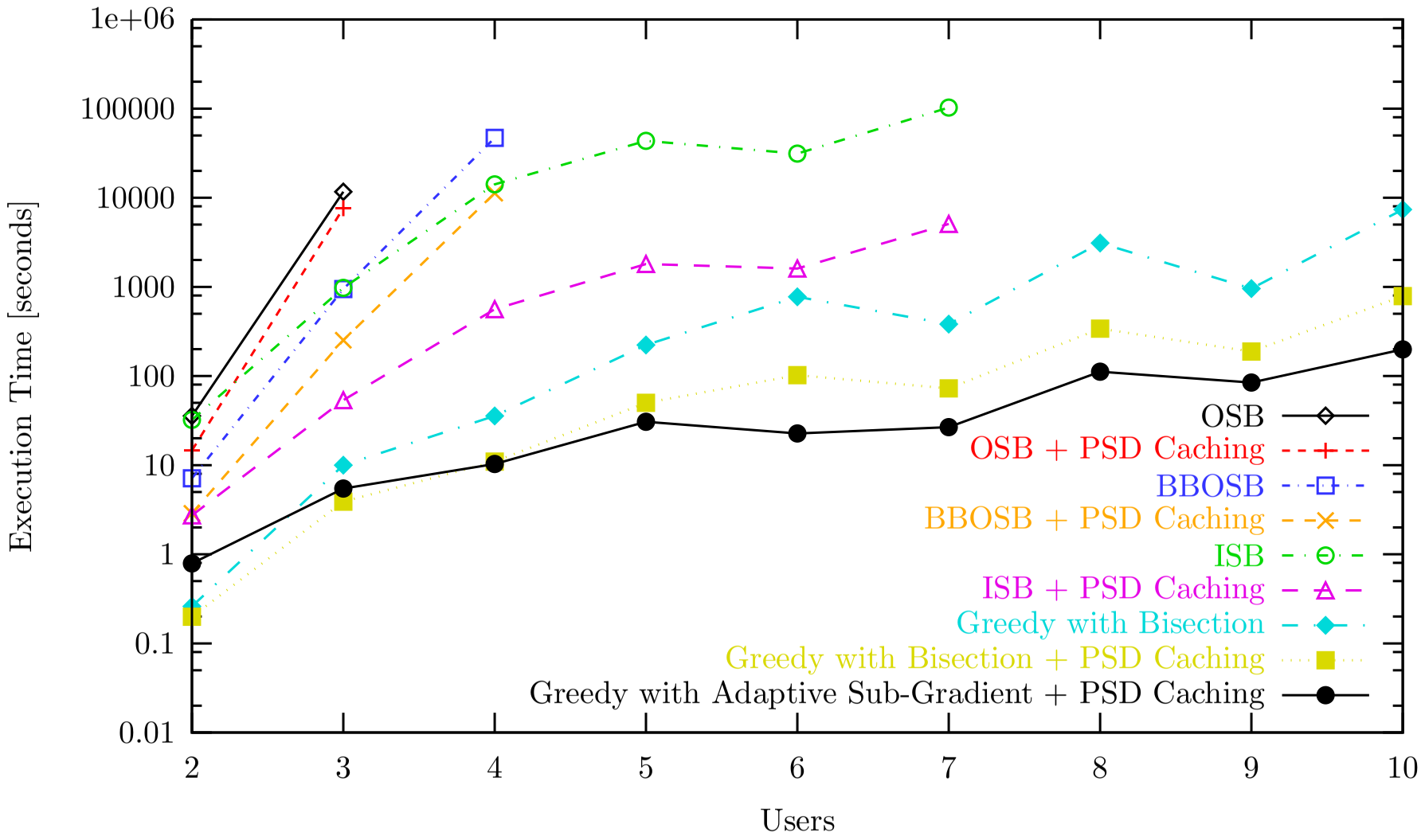


Figure 4.24: Illustration of DSM execution times from Table 4.5

Chapter 5

Conclusions

This thesis is primarily concerned with level 2 Dynamic Spectrum Management (DSM) algorithms for Digital Subscriber Lines (DSL). In Chapter 2, it was shown that currently available DSM techniques suffer from a variety of problems which make them problematic or unrealistic to utilise in today's DSL networks. In Chapters 3 and 4, a number of new algorithms for level 2 DSM were introduced, with the goal of making level 2 DSM practically realisable.

In Chapter 3, the development of a new level 2 DSM algorithm was detailed from its conception to design, known as Multi-User Incremental Power Balancing (MIPB). Later in this chapter, it was shown through extensive simulation results that MIPB produces excellent results comparable to the optimal result given by Optimal Spectrum Balancing (OSB) with equal line weights. It was also shown that the computational complexity of MIPB is significantly lower than that of other DSM level 2 algorithms and that it is tractable for large bundle sizes and was tested for up to 48 ADSL2+ lines. Up until now, no DSM level 2 results for more than 10 lines have appeared in literature. Parallelisation of MIPB was also investigated and it was discovered that excellent speedups could be obtained for up to 16 CPUs.

In Chapter 4, the Multi-User Greedy algorithm was revisited and it was found that with an adequate rate search algorithm, the Multi-User Greedy algorithm could produces near-optimal results in a range of DSL scenarios and outperform

other DSM algorithms such as Iterative Spectrum Balancing (ISB) and Iterative Water-Filling (IWF). A number of new rate search algorithms were developed, including one that can exploit parallel computation, and their relative performance assessed through simulation. A PSD vector caching technique was shown to vastly reduce the execution time of the greedy algorithm. To conclude this chapter, a summary of the absolute execution times for various algorithms was presented and it was shown that a very large reduction in computational complexity over other algorithms is obtained.

In comparison to ISB for 7 ADSL2+ users, the Greedy Algorithm with an Adaptive Sub-Gradient rate search and PSD vector caching was found to produce a solution 3850 times faster for a set of arbitrary rate targets.

Previous work on greedy algorithms which lead to the developments in this thesis were published in [15] and [16]. A patent application is currently underway based on the work in Chapters 3 and 4.

The work in this thesis goes some way to making practical level 2 DSM possible. With a fast enough level 2 DSM algorithm it is possible achieve:

- Data rate increases over IWF and SSM methods
- Power savings in the DSL bundle (i.e "Green-DSL")
- Rate adaptive DSM which further improves utilisation of the copper bundle by adapting to users bandwidth requirements

5.1 Future Work

The primary focus of this thesis has been the development of practical level 2 DSM algorithms. While there has been significant progress made in this area, there is still some distance to go before these algorithms can truly be executed in "real time". Although a parallel rate search algorithm was developed in this thesis, there is still much parallelism within the Multi-User Greedy algorithm which can be exploited to speed up the execution significantly. The obvious targets for this are parallelisation of the Δp update step which could be executed

on multiple cores in parallel. Also, the computations inside the Δp update step itself may be parallelised. Given that this step involves the solution of a large linear system in a large DSL bundle, this could be heavily parallelised and it an excellent candidate for execution on a GPU core.

If a "real time" level 2 DSM algorithm can be developed, it opens up the possibility of rate adaptive DSM to be implemented as illustrated in Section 1.2. In the rate regions show in Chapter 4, it can be seen how the rates of individual users can be traded off against one another. With a fast level 2 DSM algorithm in place, the rates of individual DSL lines could be traded against each other in response to the real time data rate requirements of each user, allowing faster peak rates as the applications demand it. In such a case, an algorithm which decides the real time rate requirements of each user would need to be developed.

References

[1] Thomas Starr, John M. Cioffi, and Peter J. Silverman. *Understanding Digital Subscriber Line Technology*, pages 67–80. Prentice Hall, 1999.

[2] P. Tsiaflakis, J. Vangorp, M. Moonen, and J. Verlinden. A low complexity branch and bound approach to optimal spectrum balancing for digital subscriber lines. In *GLOBECOM - IEEE Global Telecommunications Conference*, Nov. 2006.

[3] J. Lee, J.M. R.V. Sonalkar, and Cioffi. A multi-user power control algorithm for digital subscriber lines. *IEEE Communications Letters*, 9(3):193–195, March 2005.

[4] K. Stordahl. Broadband demand and the role of new technologies. *The 13th International Telecommunications Network Strategy and Planning Symposium*, 2008.

[5] J.M. Cioffi, H. Zou, A. Chowdhery, W. Lee, and S. Jagannathan. Greener copper with dynamic spectrum management. In *GLOBECOM - IEEE Global Telecommunications Conference*, pages 5697–5701, Dec. 2008.

[6] P. Tsiaflakis, Y. Yib, M. Chiangc, and M. Moonen. Green dsl: Energy-efficient dsm. In *IEEE International Conference on Communications*, Jun. 2009.

[7] T. Nordstrom, D. Statovci, and M. Wolkerstorfer. Energy efficient power back-off management for vdsl2 transmission. In *17th European Signal Processing Conference*, pages 2097–2101, Aug. 2009.

[8] Wei Yu, G. Ginis, and J.M. Cioffi. Distributed multiuser power control for digital subscriber lines. *IEEE Journal on Selected Areas in Communications*, 20(5):1105–1115, Jun. 2002.

[9] J. Huang, R. Cendrillon, M. Chiang, and M. Moonen. Autonomous spectrum balancing for frequency selective interference channels. pages 610–614, July 2006.

[10] W. Lee, Y. Kim, M.H. Brady, and J.M. Cioffi. Band-preference dynamic spectrum management in a dsl environment. *GLOBECOM - IEEE Global Telecommunications Conference*, pages 1–5, 2006.

[11] G. Ginis and J.M. Cioffi. Vectored transmission for digital subscriber line systems. *IEEE Journal on Selected Areas in Communications*, 20(5):1085–1104, Jun. 2002.

[12] F. Lindqvist, N. Lindqvist, B. Dortschy, P. Ödling, P.O. Börjesson, K. Ericson, and E. Pelaes. Crosstalk channel estimation via standardized two-port measurements. *EURASIP Journal on Advanced Signal Processing*, 2008:1–14, 2008.

[13] N. Papandreou and T. Antonakopoulos. Far-end crosstalk identification method based on channel training sequences. *IEEE Transactions on Instrumentation and Measurement*, 54(6):2204–2212, Dec. 2005.

[14] C.Zeng, C. Aldana, A.A Salvekar, and J.M. Cioffi. Crosstalk identification in xdsl systems. *IEEE Journal on Selected Areas in Communications*, 19(8):1488–1496, Aug. 2001.

[15] A. McKinley and A. Marshall. A new penalty based algorithm for multi-user spectrum balacing in xdsl networks. In *The 13th International Telecommunications Network Strategy and Planning Symposium, NETWORKS*, pages 1–23, Oct. 2008.

[16] A. McKinley and A. Marshall. Near-optimal multi-user greedy bit-loading for digital subscriber lines. In *Third International Conference on Access Networks*, pages 224–239, October 2008.

[17] R. Cendrillon, M. Moonen, J. Verlinden, T. Bostoen, and W. Yu. Optimal multiuser spectrum management for digital subscriber lines. In *IEEE International Conference on Communications*, volume 1, pages 1–5, 2004.

[18] Sumanth Jagannathan. *Interference and outage optimization in multi-user multi-carrier communication systems*. PhD thesis, Stanford University, 2008.

[19] Thomas Starr, John M. Cioffi, and Peter J. Silverman. *Understanding Digital Subscriber Line Technology*. Prentice Hall, 1999.

[20] P. Golden, H. Dedieu, and K. S. Jacobsen. *Fundamentals of DSL Technology*. Auerbach Publications, 2006.

[21] ANSI. Spectrum management for loop transmission systems. ANSI Standard T1.417-2003, September 2003.

[22] R. Baldemair, M. Horvat, and T. Nordstrom. Proposed method of crosstalk calculations in a distributed environment. ETSI ETC TM6 Plenary Meeting, 2003.

[23] Alcatel-Lucent, Adtran, and Conexant Systems Inc. Crosstalk channel modeling : detailed analysis and proposal. ANSI Contribution NIPP-NAI-2007-157R1, November 2007.

[24] AT&T, Adtran, and Conexant Systems Inc. 100x100 fext coupling matrix. ANSI Contribution NIPP-NAI-2007-010R2, November 2007.

[25] Alcatel-Lucent, Adtran, and Conexant Systems Inc. Revised 100 pair channel model with asymmetry. ANSI Contribution NIPP-NAI-2007-183, November 2007.

[26] J. A. C. Bingham. Multicarrier modulation for data transmission: an idea whose time has come. *IEEE Communications Magazine*, 28(5):5–14, May. 1990.

[27] J. M. Cioffi. A multicarrier primer. T1E1.4 contribution number 91-157, 1991.

[28] C.E. Shannon. Mathematical theory of communication. *Bell System Technical Journal*, 27(3-4):379–423, 1948.

[29] K.J. Kerpez. Near-end crosstalk is almost gaussian. *IEEE Transactions on Communications*, 41(5):670 –672, may 1993.

[30] F. Sjoberg, M. Isaksson, R. Nilsson, P. Odling, S.K. Wilson, and P.O. Borjesson. Zipper: a duplex method for vdsl based on dmt. *IEEE Transactions on Communications*, 47(8):1245–1252, Aug. 1999.

[31] C.M. Akujuobi, J. Shen, and M.N.O. Sadiku. A new parallel greedy bit-loading algorithm with fairness for multiple users in a dmt system. *IEEE Transactions on Communications*, 54(8):1374–1380, Aug. 2006.

[32] Thomas Starr, Peter Silverman, John Cioffi, and Massimo Sorbara. *DSL Advances*, pages 79–82. Prentice Hall, 2003.

[33] ATIS Committee. Dynamic spectrum management technical report. ATIS-PP-0600007, May 2007.

[34] R. Cendrillon and M. Moonen. Iterative spectrum balancing for digital subscriber lines. In *IEEE International Conference on Communications*, volume 3, pages 1937–1941, 2005.

[35] J. Papandriopoulos and J.S. Evans. Low-complexity distributed algorithms for spectrum balancing in multi-user dsl networks. *IEEE International Conference on Communications, ICC*, 7:3270–3275, Jun. 2006.

[36] Wei Yu and R. Lui. Dual methods for nonconvex spectrum optimization of multicarrier systems. *Communications, IEEE Transactions on*, 54(7):1310–1322, July 2006.

[37] Paschalis Tsiaflakis, Jan Vangorp, Marc Moonen, and Jan Verlinden. A low complexity optimal spectrum balancing algorithm for digital subscriber lines. *Signal Process.*, 87(7):1735–1753, 2007.

[38] Jungwon Lee, R.V. Sonalkar, and J.M. Cioffi. Multi-user discrete bit-loading for dmt-based dsl systems. In *GLOBECOM - IEEE Global Telecommunications Conference*, volume 2, pages 1259–1263 vol.2, Nov 2002.

[39] J. Lee, R.V Sonalkar, and J.M. Cioffi. Multiuser bit loading for multicarrier systems. *IEEE Transactions on Communications*, 54(7):1170–1174, July 2006.

[40] Raphael Cendrillon. *Multi-User Signal And Spectra Coordination For Digital Subscriber Lines*. PhD thesis, Katholieke Universiteit Leuven, 2004.

[41] Sumanth Jagannathan and John M. Cioffi. Distributed adaptive bit-loading for spectrum optimization in multi-user multicarrier systems. *Physical Communication*, 1(1):40 – 59, 2008.

[42] ITU. Very high speed digital subscriber line 2, itu-t recommendation g. 993.2. Technical report, ITU, 2006.

[43] V. Oksman and R. Stolle. Proposal for vdsl2 band-plan for profiles 17a and 30a. ANSI Contribution NIPP-NAI-2006-011R1, Jan. 2006.

[44] Thomas Starr, Peter Silverman, John Cioffi, and Massimo Sorbara. *DSL Advances*, pages 71–75. Prentice Hall, 2003.

[45] S. Graham, P. Kessler, and M. McKusick. gprof: a call graph execution profiler, 1982.

[46] P. Henkel. C++ threadpool framework built upon boost::thread. http://threadpool.sourceforge.net/.

[47] ITU. Physical layer management for digital subscriber line transcievers. ITU Std. G.997.1, 2003.

[48] M. Galassi et al. Gnu scientific library reference manual 3rd. edition, January 2009.

[49] G. Guennebaud and B. Jacob et al. Eigen2 library documentation v 2.0.5, 2009.

[50] A. McKinley and A. Marshall. An optimal multi-user, multi-service algorithm for dynamic spectrum management in dsl. In *Telecommunications, 2008. ICT 2008. International Conference on*, pages 1–5, June 2008.

Appendix A

A.1 ADSL2+ rate targets

This section contains tables of the rate targets used in bits per frame for the downstream ADSL2+ scenarios throughout the latter sections of Chapter 4. In each case, the target rate of line 1 is not set and is therefore maximised with all the the DSM algorithms tested.

Line 2	2000	8 Mbps

Rate targets for 2-User ADSL2+ downstream scenario

Line 2	2000	8 Mbps
Line 3	3200	12.8 Mbps

Rate targets for 3-User ADSL2+ downstream scenario

Line 2	2000	8 Mbps
Line 3	3200	12.8 Mbps
Line 4	4000	16 Mbps

Rate targets for 4-User ADSL2+ downstream scenario

Line 2	2000	8 Mbps
Line 3	3200	12.8 Mbps
Line 4	4000	16 Mbps
Line 5	3900	15.6 Mbps

Rate targets for 5-User ADSL2+ downstream scenario

Line 2	1800	7.2 Mbps
Line 3	3300	13.2 Mbps
Line 4	3800	15.2 Mbps
Line 5	5400	21.6 Mbps
Line 6	2600	10.4 Mbps

Rate targets for 6-User ADSL2+ downstream scenario

Line 2	1800	7.2 Mbps
Line 3	3300	13.2 Mbps
Line 4	3800	15.2 Mbps
Line 5	5400	21.6 Mbps
Line 6	2600	10.4 Mbps
Line 7	1600	6.4 Mbps

Rate targets for 7-User ADSL2+ downstream scenario

Line 2	1600	6.4 Mbps
Line 3	3200	12.8 Mbps
Line 4	3600	14.4 Mbps
Line 5	5200	20.8 Mbps
Line 6	2400	9.6 Mbps
Line 7	1400	6.4 Mbps
Line 8	1000	4 Mbps

Rate targets for 8-User ADSL2+ downstream scenario

Line 2	1600	6.4 Mbps
Line 3	3200	12.8 Mbps
Line 4	3600	14.4 Mbps
Line 5	5200	20.8 Mbps
Line 6	2400	9.6 Mbps
Line 7	1400	6.4 Mbps
Line 8	1000	4 Mbps
Line 9	2000	8 Mbps

Rate targets for 9-User ADSL2+ downstream scenario

Line 2	1600	6.4 Mbps
Line 3	3200	12.8 Mbps
Line 4	3600	14.4 Mbps
Line 5	5200	20.8 Mbps
Line 6	2400	9.6 Mbps
Line 7	1400	6.4 Mbps
Line 8	1000	4 Mbps
Line 9	2000	8 Mbps
Line 10	1500	6 Mbps

Rate targets for 10-User ADSL2+ downstream scenario

Appendix B

B.1 Architecture of DSL Bit-Loading Simulation

The DSM simulation tool used in this thesis was developed from scratch in C++. The channel models and crosstalk models utilised are detailed in sections 2.3 and 2.4. The GNU Scientific Library [48] was utilised for the solution of the linear system in equation 2.25 via LU decomposition.

During development the use of Eigen2 [49] (an advanced C++ linear algebra library with automatic vectorisation for SSE instructions) was also attempted. The Eigen2 library uses a partial pivoting LU decomposition which is faster but less numerically stable than that of a fully pivoting LU decomposition. It was discovered that the BBOSB algorithm would not function correctly using this library as it relies heavily on numerical stability of the upper and lower bound calculations.

The parallelisation of the DSM algorithms investigated was undertaken using posix threads and the *boost::threadpool* [46] library. The PSD vector cache uses a per tone lock (a pthread_mutex_t) to ensure that concurrent accesses from separate threads do not cause corruption of the cache.

The accuracy of the simulations were verified by comparing simulation results for OSB with a data set generated by Paschalis Tsiaflakis from Katholieke Universiteit Leuven.

Appendix C

C.1 List Of Publications

- A. McKinley and A. Marshall. An optimal multi-user, multi-service algorithm for dynamic spectrum management in dsl. In *Telecommunications, 2008. ICT 2008. International Conference on*, pages 1–5, June 2008

- A. McKinley and A. Marshall. Near-optimal multi-user greedy bit-loading for digital subscriber lines. In *Third International Conference on Access Networks*, pages 224–239, October 2008

- A. McKinley and A. Marshall. A new penalty based algorithm for multi-user spectrum balacing in xdsl networks. In *The 13th International Telecommunications Network Strategy and Planning Symposium, NETWORKS*, pages 1–23, Oct. 2008

- U.K. patent application - "Rate Adaptive Bit-Loading in DSL Bundles"