ε-approximate Algorithm for SLA Negotiation

Basem Shihada
bshihada@bbcr.uwaterloo.ca

Mohamed Soliman
msoliman@bbcr.uwaterloo.ca

July 25, 2002

Abstract

A Service level agreement (SLA) requires the ISP to provide high reliability, customer satisfaction, and as a result can gain high profit. The current implementations of SLA perform negotiation operations on the ISP side, since the ISP usually provides certain service package. It is the responsibility of the customer to analyze the available packages then choose his preference over ISPs. This could result in some ISP's gaining many major opportunities for business contracts. Multi-objective optimization approach finds all possible solutions to meet the objectives of each party (client, ISP). It offers a set of solutions "Pareto-set" that results in a more flexible and optimal set of solutions to provide multiple SLAs with different conditions that satisfy the client's criteria.

This project shows a novel approach to finding a set of solutions for SLAs as a multi-objective optimization problem using an ε-approximation algorithm provided by [3]. The goal of this design is to formulate the SLA problem and to develop the required set of solutions using approximation algorithms. The project discusses the algorithmic complexity of SLA negotiation and the required implementation of the algorithm proposed by [3]. The use of [3] algorithm provides several enhancements to SLA solutions, such as finding solutions, which can be mapped over Pareto-curve instead of solutions generated form the utility function. The achieved solutions will be close to the optimal solution results by a factor of ε, and also in lower complexity.

1 Introduction

The Internet has been traditionally perceived as a best-effort network. Delivering Internet services to customers relied on flat pricing schemes. For example, certain hours of usage or unlimited usage per month was provided for a fixed price. Because of the success and the enhanced availability of the Internet, businesses have begun to rely on it by constructing private networks over it (Virtual Private Networks). Thus saving the cost of leasing lines or building their own networks. This has also been seen recently in the focus on using Web Services, B2B, B2C,..etc. These classes of new applications demanded new requirements for Internet access with quality of service guarantees (QoS). Therefore, traditional pricing schemes are now categorical. Research efforts have emerged to provide different classes of QoS while providing guarantees to the customers.

1.1 Service Level Agreements (SLA)

Service level agreements (SLA) are the means for contracting between the service provider, e.g. ISP and the customer. One part of the SLA is performance guarantees; such

as delay, jitter, and availability. Because of the above-mentioned reasons, incorporating SLAs have gained recent attention as a mean for the customer to monitor the promises made by the ISP and a mean for the ISP to deliver the appropriate level of performance. From the perspective of the customer, there are several ISPs to choose from and from the ISP perspective there are many customers to satisfy with different levels of QoS demands.

1.2 Problem Set

SLA negotiation is about the delivery of the performance criteria defined by the customer (Service Level Objectives, thresholds). The customer's objectives are defined to maximize the delivered performance with minimum cost. From the Service Provider (SP) perspective, it is important to maximize profits and to allocate resources efficiently. The SP has the objective of maximizing profits according to a set of constraints defined by the customer. This problem is a multi-objective optimization problem, which is hard to solve.

The availability metric is the most commonly demanded parameter in SLAs. Availability is the percentage of time the service is available. To motivate businesses to rely on the Internet, higher levels of availability need to be achieved. Although 99.8% availability is hard to achieve, as the Internet's best effort, 99.8% is not quite good for businesses. It implies 1.75 hours of mean accumulated down time per month [1]. Most Service providers are struggling to achieve the six nines figure (99.9999%), which translates to 2.6 seconds per month.

In order to commit to the promised performance, the service provider can receive a price p for the fulfilled service level and has to pay a penalty l for any unfulfilled promise. Obviously the SP has no incentive to report on the undelivered service levels. SLA monitoring is about monitoring and measurement of the delivered service to verify the contract. Unfortunately, SLA monitoring does not come with zero cost. This is due to added listeners, which add more traffic and load to resources. A Long reporting interval leads to a high risk and a small reporting interval has an extra cost. The equilibrium on the frequency of reporting is a trade-off between the risk and the cost of reporting.

1.3 Paper Outline

This project presents common SLA negotiation problems viewed as a multi-objective optimization problem. It will particularly highlight an approximation algorithm which leads to generate a set of solutions close to optimal (Pareto-set) by a factor of ε.

The rest of the paper is organized as follows: section 2 provides a literature survey of the available solving techniques for multi-objective optimization problems. Section 3 analyzes the SLA problem. Furthermore, it describes the theoretical model of SLA and the current solving mechanisms. Section 4 specifically analyzes Pareto curve complexity and why an approximation is needed. Section 5 illustrates SLA problem formulation based on SLA's criteria, and objectives. Section 6, provides a detailed formulation for ISP reliability proposing certain home networks, which have multiple ISPs and clients.

Section 7 focuses on how to apply ε-approximation algorithm [3] to SLA negotiation problem. Furthermore, it provides a detailed algorithm analysis (complexity). Section 8 concludes with the goals achieved from this project. Section 9 explores the opportunities for future extensions. Finally, Appendix A some implementation results.

2 Multi-objective Optimization Solutions

Multi-objective problems are different from single objective problems. In single objective problems, the objective is to maximize or minimize a certain objective function according to a set of constraints. Taking a linear objective function and a linear set of constraints, the simplex method can be used to solve this problem. Several algorithms for the linear programming problems can achieve polynomial time complexity.

The first SLA problem mentioned earlier (profit maximization and customer satisfaction) is a problem of multi-objective optimization. When we have multiple objectives, we usually do not have a single solution that is the best for all objectives. Instead, we have multiple solutions with each one corresponding to be better for certain objective(s) and worse to others. Therefore, solutions cannot be easily compared with each other. This is because objectives don't meet on everything and conflicts may exist. Such solutions are called *non-dominated* solutions or *Pareto optimal* solutions [5], which capture tradeoffs among objectives.

2.1 Evolutionary Algorithms

Genetic Algorithms are forms of evolutionary approaches, which provide success for the multi-objective optimization problem [6,8,9,16]. A genetic algorithm generally consists of the following five steps. "First, A genetic representation of solutions to the problem. Second, a way to create an initial population of solutions. Third, an evaluation function rating solutions in terms of fitness. Fourth, genetic operators rating solutions in terms of their fitness. Fifth, values of the parameters of genetic algorithms" [6].

Some interesting issues are in regards to theoretical aspects of evolutionary algorithms, summarized as:
a) *Problem hardness* [9]. It distinguishes what is hard versus what is an easy problem for the evolutionary algorithm.
b) *Computational complexity* of algorithms themselves. As indicated in [9], there have not been concrete results on the computational complexity of evolutionary algorithms for non-trivial problems although the complexity theory is well established.
c) The ergodic convergence property of genetic algorithms defines that when the search time goes to infinity, the number of global minima achieved will be several. Other approaches will achieve one. That gives an indication for a good performance of genetic algorithms in convergence.
d) *The No Free Lunch Theorem* [11]. This theorem claims that all optimization techniques have similar behavior on average, given that we do not know about the objective functions. This theory was argued in a paper titled " Perhaps not a Free Lunch

but an Appetizer" [12]. In this paper, the authors argued why optimization techniques differ in performance for realistic scenarios.

2.2 Nash Equilibrium

So far, we have considered optimizing the multi-objective problem, while assuming only one decision maker. When multiple decision makers are involved, different views may exist. Cooperative versus non-cooperative games studied these issues from the point of game theory. Nash-Equilibrium is a well-established concept from the pioneering work by John Nash[1]. Nash equilibrium is the point where no one else can improve his utility[2] function. An interesting question is how to define the incentive compatibility among the different entities and to maximize them. In trying to optimize for multi-criteria, Nash-Pareto equilibrium heads in that direction [10].

2.3 ε-approximate Pareto Curves

Pareto optimal solutions is a common notion for multi-objective optimization problems because it captures the tradeoff among objectives, which is exponential in size. Even for two objectives, determining whether a point belongs to the Pareto curve *P(x)* is NP-hard. Figure 1 shows Pareto curve [3] for minimizing two objective functions. Moreover, dynamic SLAs may involve several iterations that increase the complexity of the problem and require faster response time. Therefore, we are looking for reducing the time complexity in negotiating and establishing an SLA.

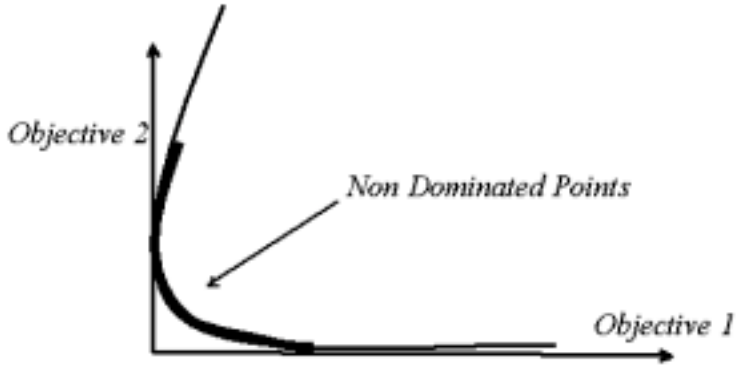

Fig.1 Pareto Optimal Solutions for two objectives (Min-Min), *P(x)*

ε-approximation algorithms are designed to have an input ε in addition to the problem. The ε factor is called the approximation ratio or the relative error bound [2]. A polynomial-time approximation scheme is for any fixed ε >0, the algorithm runs in time polynomial with size n of the input instance. The fully polynomial-time approximation scheme (FPTAS) is used when the running time is polynomial both in 1/ε and in the size

[1] Holder of Nobel Prize in Economics, 1994.
[2] A utility function is a function of decision variables.

of *n* of the input instance [2]. For example, if ε is doubled, the running time should not increase exponentially but to be decreased with the same factor (half).

Based on the work done by Papadimitriou [3] for approximating multi-objective problems, we can find a polynomial time approximation scheme for finding Pareto solutions for the SLA problems mentioned above. As defined in [3], the ε-approximated Pareto curve $P_\varepsilon(x)$ can be defined as a set of solutions, which are not dominated by any other by a ratio of more than 1+ε. At least the first two theorems in [3] are important to us.

For optimization problems are such that "if f(x,s) > 0 then f(x,s) is between $2^{-p(|x|)}$ and $2^{p(|x|)}$ *and for some polynomial p; this is a consequence of the polynomial nature of the solutions and the objective function f where x is problem instance, s is a feasible solution, f is the required algorithm, and we need arg max f(x,s)"*[3].

*"**Theorem 1** For any multi-objective optimization problem and any* ε *there is a* $P_\varepsilon(x)$ *consisting of a number of solutions that is polynomial in |X| and 1/ε (but exponential in the number of objectives)"* [3].

*"**Theorem 2** There is an algorithm for constructing a* $P_\varepsilon(x)$ *polynomial in |x| and 1/ε if and only if the following problem can be so solved: Given x and a k-vector* $(b_1, \ldots, b_k)$, *either return a solution s with* $f_i(x,s) \geq b_i$ *for all i, or answer that there is no solution s' with* $f_i(x,s') \geq b_i(1+\varepsilon)$*"*[3].

The first theorem is about the existence of an approximate Pareto curve for a certain class of objective functions. The second theorem is about the existence of an algorithm for finding an approximate Pareto curve under specific conditions.

3 SLA Problem Analysis

SLA negotiation is multi-objective in nature. In a single objective problem, the objective is to maximize or minimize a certain objective function according to a set of constraints. Taking a linear objective function and a linear set of constraints, the simplex method can be used to solve this objective function. Linear programming algorithms can solve this kind of problem. SLA negotiation problems consist of profit maximization and customer satisfaction from the ISP side, combined with customer maximization of penalty and satisfaction from the client side. In this case SLA negotiation took a multi-objective optimization in nature.

One possible solution to this problem is to use the notion of a utility function [1]. Utility functions combine weighted combinations of objectives. It results in finding one optimal solution. The drawbacks of minimizing or maximizing weighted convex combinations of objective functions are analyzed by [4]. For example, the SLA negotiation problem is characterized by the need to minimize or maximize several non-linear functions of the constraints simultaneously. The use of a utility function can only help encode each objective preference to be a single objective, but failed to generate a set of solutions to

multi-objective problems. Since a utility function finds solutions based on weighted convex combinations it is quite impossible to find the correct weights needed to generate the optimal solution [4]. A conclusion provided by [4] is that finding the correct weighted convex combinations is the major obstacle since it is impossible to know the correct weights without knowing the shape of Pareto curve.

Pareto solutions outperform the utility function approach by providing multiple optimal solutions rather than one. They are, as previously mentioned, the set of non-dominated solutions.

4 Pareto Curve Complexity Analysis

Any optimization problem is considered by [3] as a set of *instance*s, *solutions,* and *objective functio*ns. As mentioned earlier, in multi-objective optimization we are not interested in a single optimal solution, but in capturing the notion of a "trade-off" called "*Pareto solutions".* Pareto curve captures the notion of a "trade-off."

Constructing Pareto curve is hard; Since Pareto curve has an exponential size. Even for fewer of objectives e.g. two objectives, determining if a point belongs to P(x) is NP-hard [3]. One of the possible ways to study Pareto curve solutions is to study the possibilities of approximating Pareto curve. In other words, determine the set of solutions, which are not dominated by any other by a small ratio ε. It has been shown in [3] that, under very general conditions, there is a polynomial time approximate Pareto curve by ε. ε-approximate Pareto curves have not been investigated by several work in computationally-oriented multi-objective optimization literature [9]. However, with the exception of certain straightforward results in [3], the complexity issues of finding Pareto raised have not been addressed systematically.

A polynomial in size "ε-approximate Pareto curve always exists, but generally it is hard to construct" [3]. It also provided a sufficient condition for the existence of an approximate Pareto solution. The above results could be applied to SLA negotiation multi-objective optimization problem.

5 SLA Problem Formulations

To scale the SLA negotiation problem, it must be formulated in such a way that the number of objective functions are kept to a minimum. Otherwise, the problem cannot scale using any available multi-objective algorithm. SLA problem formulation is divided into three major threads, SLA constraints, SLA goals, SLA objectives.

One candidate constraint is packet delay, in which this measurements aim to assess the network performance in order to know what quality is available which supports various types of customer services. Another example is Jitter, which is a result of the deviation in or displacement of some aspects of the packet transmission ordering. In other words, transmission jitter is the short-term variation of the significant instants of packets from their ideal positions in time. Security constraints could take place if the customer is

running a highly secure application and wants to make sure that the packets are transmitted through trusted nodes with some level of security monitoring management. SLAs need to take the available constraints and make sure their performance fulfills the agreement. SLA goals are to surpass certain thresholds as much as possible. For example, packet delay must be < 50 ms and Reliability > 99.8 with the best improvement. SLA goals (SLG) can be converted into a set of objectives (goal programming) [16]. On the client side, satisfaction and service penalty payment are very important. Usually client objectives are to maximize their satisfaction and to maximize the penalty payment in case the ISP failed to provide the service, which has been already agreed on.

If the service level is not achieved, then either party wants to find the closest possible solution. As a result, the programming goal can be converted into a Multi-Objective Problem (MOP) [16]. The next section will assume a home network and how to formulate the SLA negotiation between clients and ISPs.

6 Reliability Formulation

In order to increase service reliability, a backup SP could be in service in the case that the first one fails. If the first one has a probability of failure p_1 and the second SP has a probability of failure p_2, and assuming independence of failures, the resultant reliability is significantly enhanced. The resultant probability of failure will be $(1-p_1*p_2)$. This concept can be generalized to use the multi-home network. In a multi-home network, the resultant service is a combination of agreement with n service providers. Figure one illustrates a home network, which consists of multiple ISPs and multiple clients. A centralized ISP (home) is trying to provide a service to the clients such that the reliability is ≥ the six nines figure. Assuming the probability of each ISP to fail is *p*, penalty on failure *l*, and cost of service is *c*. The problem is formulated as to find *n* such that:

Minimize (Penalty $F_l(n)$),
Maximize (Reliability $F_R(n)$)
Minimize (Cost Fc(n))

where,

$F_l(n) = \sum ({}^nC_i * p^i * i * l)$, i from 1 to n
$F_R(n) = (1-p^n) * 100$ Assuming independence of failure
$F_c(n) = n * c$

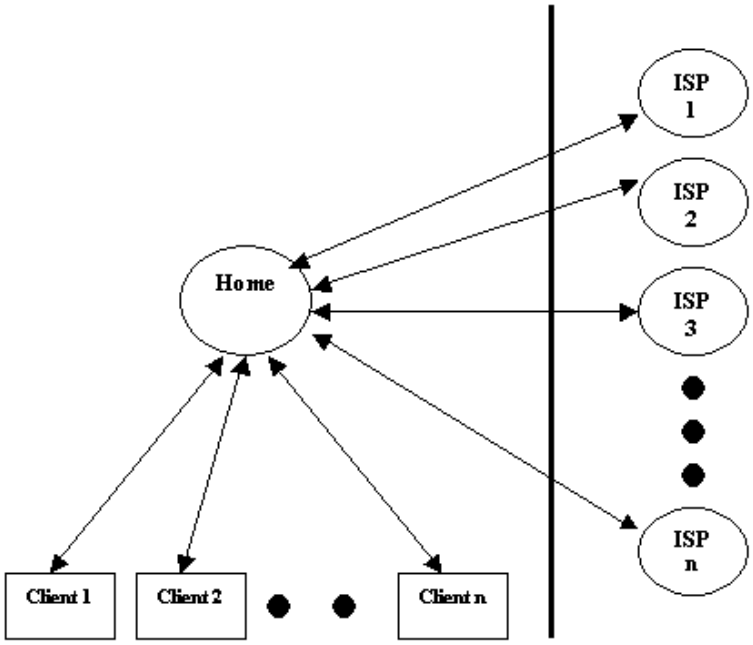

Fig. 2 SLA Home Network

7 PTAS Algorithm for Pareto Approximation

The solution to finding a near optimal Pareto curve could be found by iterating through the utility function by varying weights to generate a set of Pareto-close solutions similar to the algorithm proposed by [3]. That could be better than calculation of the utility function as Max ($\Sigma w_i v_i$), ($\Sigma w_i = 1$), which generates one optimal solution.

Algorithm1 [3]

Given *x;*
for each vector $w \in \{0,\ldots..,M\}^k$ do
    Find the optimum of *x* under objective $v = \Sigma_i w_i v_i$
    Return all optima thus found
Loop

"It is not hard to generate all optima returned that are in P(x). Also it is always possible to loop over all positive weighted values, instead of these bounded integer values, then all of P(x) would be recovered. By restricting the algorithm to small integer weighted values it is intuitive that the algorithm is going in the right direction of the real P(x) solutions" [3].

Keeping in mind the above-mentioned issues, we can map the SLA negotiation problem to be a polynomially solvable problem. Algorithm 2 illustrates the steps needed to

generate approximate Pareto solutions P_ε (x). "Set $M = \lceil 4k^2/\varepsilon \rceil$, k = number of objectives. For each *i*, loop through the following values of w_i: Starting with 0, increment w_i by 1 until M; after that continue going only through the even numbers until 2M. Then go on with multiples of 4 until 4M. Continue in this manner increasing the power of 2 until $2^{2p(|x|)}$ M. For each combination of w_i's, find the optimum of *x* under a single objective $v = \sum w_i v_i$" [3].

Algorithm2 [3]

Set $M = \lceil 4k^2/\varepsilon \rceil$, k = number of objectives

For J = 0 to $2^{2p(|x|)}$ M
    $w_{i=0}$
    $\Sigma = \Phi$
    For I = 0 to k
        $w_i = w_i + 2^j$
        $\Sigma = \Sigma\ w_i v_i$
    loop
  Maximize $\Sigma_i = w_i v_i$
  Include result in $P_\varepsilon(x)$
loop

The complexity of the above algorithm has been studied in [3] to find that the number of combinations be in $O((8p(|x|)k^2/\varepsilon)^k$. Obviously, the algorithm is $1/\varepsilon$. But it is to the power of k (number of objective functions). For this algorithm to scale, it is critical to take the number of objective functions k in consideration. This algorithm can be improved by assuming that SLA negotiation objective functions are efficiently approximable [3].

8 Conclusion

In this project, several service level agreement decision problems were explored. In order to achieve dynamic SLA establishment, there is a need for automating SLA negotiation. We have mapped the SLA negotiation problem to a multi-objective optimization problem. We have tackled several approaches for solving the multi-objective optimization problem. Pareto-optimal solutions, and genetic algorithms are potential candidates. Pareto solutions are the set of non-dominated solutions that capture the trade-offs among objectives. Because of the resultant NP-hardness, we needed a polynomial time approximation algorithm. The *ε-approximate Pareto Curve* algorithm is a candidate because of the proven polynomial time approximation. The use of a utility function can help encode each objective function to result in a single objective, which is easier to solve. The utility function approach may have strengths despite its weaknesses. Multi-objective optimization is a complex and a wide research area. Finding SLA best decisions imply finding all the set of optimal solutions and to be able to rank them.

9 Work Extensions

For future considerations, it is necessary to expand the scale of the SLA model to include more thorough cost models. For the current approximation algorithms, it is important to compare the algorithm proposed by [3] in terms of time complexity and accuracy with other algorithms and approximations against the utility model. It is also important to investigate the criteria for ranking Pareto-set, in other words, selecting solutions using preferences from the resultant solutions. Also answers to questions like "is the result of an MOP is power-law like? And Why?" can be investigated and provided [5]. In order to study compatibility among objective functions, we need to study an interesting approach taken from game theory and economics called Nash-Equilibrium [10].

Acknowledgements

We would like to thank Dr. López-Ortiz for his insightful discussions, which led us to more relate SLA problems to multi-objective techniques and many other topics discussed within this seminar.

References

[1] J. Park and J. Beak, "Management of Service Level Agreements for Multimedia Internet Service Using a Utility Model," IEEE Communications Magazine, May 2001.

[2] T. Cormen, C. Leiserson, and R. Rivest, "Introduction to Algorithms," 1997, MIT press.

[3] C. H. Papadimitriou, M. Yannakakis, "The Complexity of Trade-offs and Optimal Access of Web Sources," *FOCS 2000.*

[4] I. Das, J. Dennis "A closer look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicritera Optimization Problems", Dec. 1996, available at citeseer.

[5] C. Papadimitriou "Algorithms, Games, and the Internet," Available at http://www.cs.berkeley.edu/~christos/

[6] M. Gen, R. Cheng, "Genetic Algorithms & Engineering Optimization," 2000, Wiley & Sons.

[7] R. Keeney and H. Raiffa, "Decisions with Multiple Objectives, "Preferences and Value Tradeoffs, ", John Willey and Sons, 1976.

[8] M. Sakawa, "Genetic Algorithms and Fuzzy Multi-objective Optimization," Kluwer Academic Publishers, 2001

[9] X. Yao, "Evolutionary Computation Theory and Applications", World Scientific Publishing, 1999 pp 10-11.

[10] E. Altman et al, "A survey on networking games in telecommunications", available at citeseer.

[11] D. Wolpert and W. Macready, "No Free Lunch theorems for Search", available at Citeseer

[12] Droste et al, “Perhaps not a free lunch but at least a free appetizer,” GEECO’99, Available at citeseer
[13] G. Rudolph, “Convergence of Evolutionary algorithms in general search spaces, “ ICEC’96, IEEE press
[14] Von Neumann, J. and O. Morgenstern, “Theory of Games and Economic Behavior”, 2nd ed. Princeton University Press (1947), Princeton, N.J.
[15] Z. Liu, M. Squillante and J. Wolf, “On Maximizing Service-Level-Agreement Profits,” EC’2001, ACM Digital Library.
[16] K. Deb, “Multi-Objective Optimization Using Evolutionary Algorithms, “ John Wiley & Sons (2001).

Appendix A

Experimental Results

Representative results are presented to highlight the approximation algorithm performance and the generated results. Several ε values have been selected and used to track the correctness of the generated results.

Penalty l (3), SLA cost c(4), ε value, objective functions k = 2, and *ISP probability of failure p(0.002)* are the experiment parameters.

Figure 1, 2, and 3 show the distribution of the resultant solutions between objective function one and two with different values of ε.

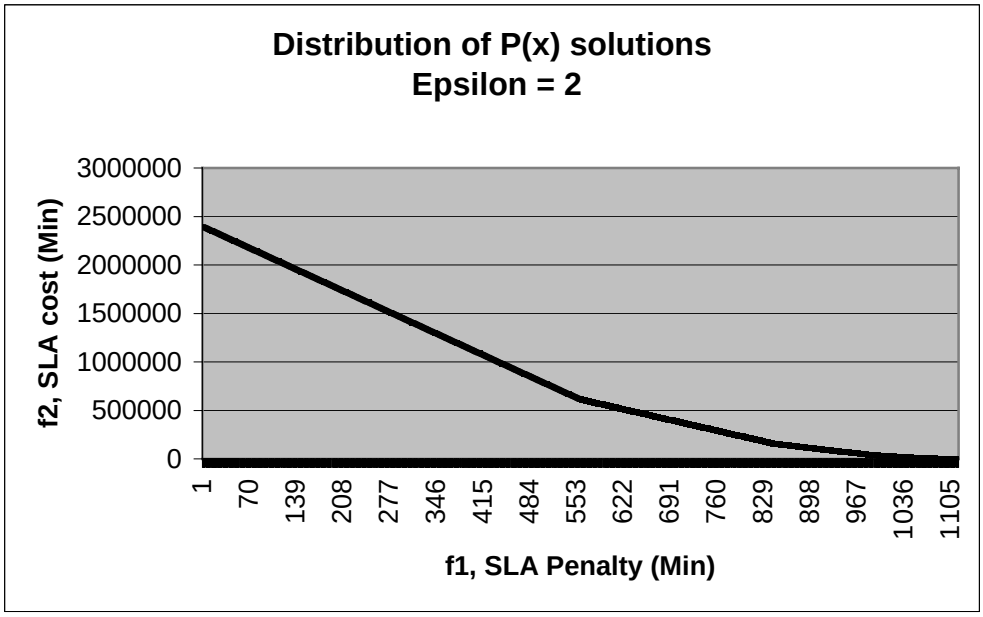

Fig. 1 Distribution of $P_{\varepsilon}(x)$ Solutions ε=2

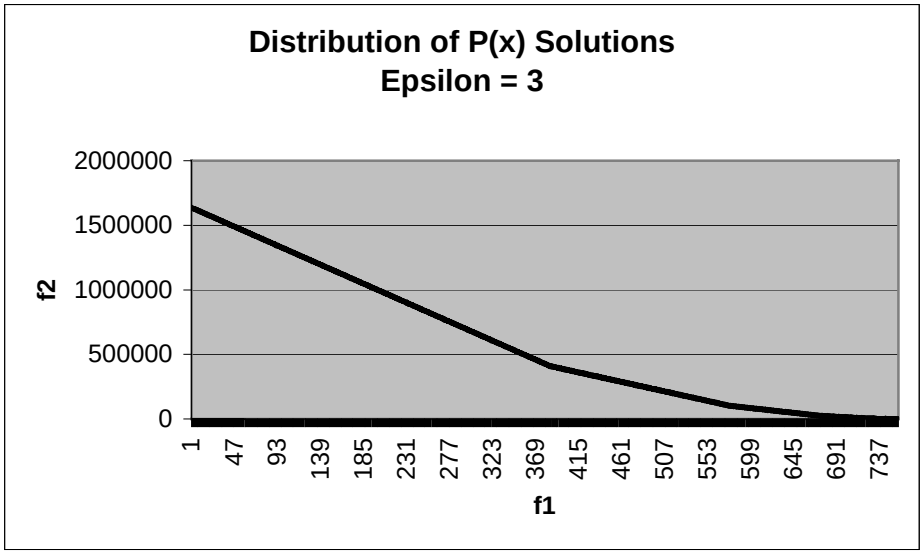

Fig. 2 Distribution of $P_\varepsilon(x)$ Solutions $\varepsilon=3$

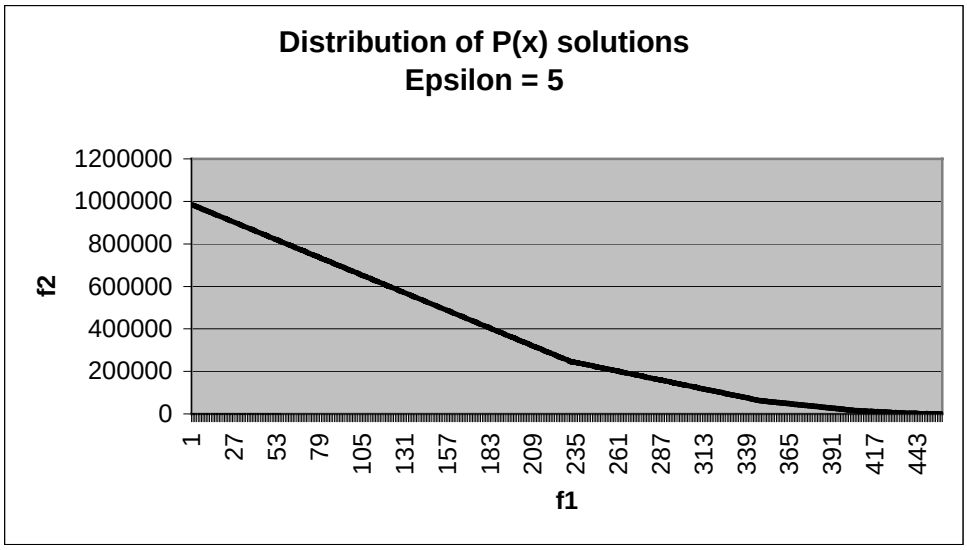

Fig. 3 Distribution of $P_\varepsilon(x)$ Solutions $\varepsilon=5$

Analysis

From the figures mentioned above, the set of solutions for higher epsilon are converging to certain direction. This means that for higher value of epsilon, the resultant curve will be closer to Pareto Curve.

Appendix B

The following is a C language implementation of an ε-approximation algorithm for the SLA problem mentioned above.

```
/*

        This is an implementation of [3] C. H. Papadimitriou
        epsilon-approximation algorithm mapped to SLA negotiation
                                problem

                                July 2002

        Bugs, questions, suggestions, additions, whatever, to

                Basem Shihada
                school of Computer Science
                University of Waterloo

                email bshihada@bbcr.uwaterloo.ca
                tel 519 888-4567 ext 6238


        This implementation is based on the pre-described ISP home network,
        the used objective factions are formulated in section 6, furthermore
        the implemented algorithm is illustrated in section 7.


*/


#include<stdio.h>
#include<math.h>
#include<lip.h>

FILE *fout1, *fout2;        // output files for P(x) solutions
int x = 0;                  //find the value of x where make function has a maximum
                            // or minimum value
float epsilon =0; // Epsilon value
int wi = 1;                 // weighted values
double M = 0 ;              // = 4K2 / epsilon
int k = 3 ;                 // number of objective functions
int i,j;            // defined for loop iteration only
float penalty;              // penalty value
float coast;                // SLA coast in $ per packet
double p = 0.002;           // probability of each ISP failure
int n = 6;                  // number of ISP
float val=0;

double f1max = 0;           // storing the summation for the generated maximum of f1
double f2max = 0;           // storing the summation for the generated maximum of f2
double f3max = 0;           // storing the summation for the generated maximum of f3
```

```
/********************* M E T H O D  H E A D E R ***********************
// Method : minf1()
/*************************************************************************
* find the minimum of the objective function one
* max Fp(n) =  summation (nCi * pi * c)   , i from 1 to n
* @param
*
* @exception
*************************************************************************/
/* Author       : Basem Shihada  - 18/7/2002
*
* Modification : Author - DD/MM/YYYY + Short description
*************************************************************************/


// compute factorial

int fact(int val)
{
   if (val == 0)
      return(1);

   return (val * fact(val-1));
}

double minf1(int wi, int i, int x)
{
int m1,m2,m3,j;
double temp;

        m1 = fact(x);
            m2 = fact(x-i);
            m3 = fact(i);

            val = val + (m1 / (m2 * m3)) * p * coast * wi;

            fprintf(fout1,"maximum of function 1 is: %d %f \n",x,val);


return val;
}

/********************* M E T H O D  H E A D E R ***********************
// Method : maxf2()
/*************************************************************************
* find the maximum of the objective function two (reliability)
* max Fr(n) = (1-pn)*100    Assuming independence of failure
* @param
*
* @exception
*************************************************************************/
/* Author       : Basem Shihada  - 18/7/2002
*
* Modification : Author - DD/MM/YYYY + Short description
*************************************************************************/
float maxf2(int wi, int x)
```

```
{
float val;

val = wi * (1- pow(p,x))*100;

fprintf(fout1,"maximum of function 2 is: %d %f \n",x,val);

return  val;
}

/********************** M E T H O D  H E A D E R ***********************
// Method : minf3()
/*************************************************************************
* find the maximum of the objective function one
* max Fc(n) =  n *  coast
* @param
*
* @exception
*************************************************************************/
/* Author       : Basem Shihada  - 18/7/2002
*
* Modification : Author - DD/MM/YYYY + Short description
*************************************************************************/
float minf3(int wi, int x)
{

fprintf(fout2,"minimum of function 3 is: %d %f \n",x, x * wi * coast);

return x * wi * coast;
}

/********************** M E T H O D  H E A D E R ***********************
// Method : main()
/*************************************************************************
* main method for Papademetrio epsilon approximation algorithm
*
* @param  Epsilon Pareto curve approximation
*
* @exception
*************************************************************************/
/* Author       : Basem Shihada  - 18/7/2002
*
* Modification : Author - DD/MM/YYYY + Short description
*************************************************************************/
int main()
{


/*
                    Some of methods has been re-used from the free Source code
                    long arithmetic library called lip.h

                    The objective functions are:

                    min Fp(n) =  å (nCi * pi * c)   , i from 1 to n
                    max Fr(n) = (1-pn)*100    Assuming independence of failure
```

```
                    min Fc(n) =  n *  w

*/

// initialize the output result file

if((fout1 = fopen("results1","w"))==NULL)
   printf("can not open the file %s,exit\n","results1");

if((fout2 = fopen("results2","w"))==NULL)
   printf("can not open the file %s,exit\n","results2");

printf ("Enter Epsilon value\n");
scanf ("%f",&epsilon);

printf ("Enter Penalty value\n");
scanf ("%f",&penalty);

printf ("Enter SLA Coast value\n");
scanf ("%f",&coast);

M = floor (4 * k * k)/epsilon ;
printf("The resultant M is %f\n",M);

for (j=0;j<n ;j++)
{
        for (i=0;i < (pow(2,j) * M);i++)
        {
                if (x<n)
                {
                  //f1max = maxf1(wi,i,x);
                     f2max = maxf2(wi,x);
                   }
                   wi = wi + (int) pow(2,j);
        }
x++;
}
x =0;
wi =0;

for (j=0;j<n ;j++)
{
        for (i=0;i < (pow(2,j) * M);i++)
        {
                if (x<n)
                {
                     f3max = minf3(wi,x);
                   }
                   wi = wi + (int) pow(2,j);
        }
x++;
}
return 0;
}
```