Lagrangean relaxation for service location in large-scale networks with QoS constraints

Zille Huma Kamal, Ala Al-Fuqaha∗,† and Ajay Gupta

Department of Computer Science, Western Michigan University, 1903 W. Michigan Ave., Kalamazoo, MI 49008, U.S.A.

Summary

Current trends in computing indicate that there is a great potential for service-oriented computing and similar technologies, such as application-oriented networks (AONs), where services can relocate to adapt to the conditions of the underlying network. In such environments, providing and consuming services and establishing a relationship between consumers (users of services) and producers (providers of services) are still challenging and vastly researched aspects.

Bearing this in mind, we define a service location and planning (SLP) problem that uniquely matches producers to consumers and accounts for realistic parameters such as, quality of service (QoS) constraints of throughput and delay, and network constraints of underlying link layer bandwidth capacities, and cost of meeting consumer requests.

Our contribution lies in the mathematical formulation of the SLP problem as an integer linear programming (ILP) problem that can be solved optimally for small-scale networks and extending this work using Lagrangean relaxation (LR) approximation techniques to solve the SLP problem for large-scale networks. Copyright © 2009 John Wiley & Sons, Ltd.

KEY WORDS: managing network resources and services; Lagrangean relaxation; service location and planning problem; application-oriented networks; opportunistic networks; optimization problem; integer linear programming problem

1. Introduction—Current Demands from Networking

The internet has evolved from a simple, restricted network allowing communication between fixed points to a mesh of global internetworking technology. In this evolution, various computing paradigms have sprouted up ranging from mobile computing to wireless sensor networking to pervasive computing to the emerging and still evolving web services and service-oriented computing (SOC).

A closer look at any of these paradigms will give insight into the direction of research for that paradigm and its demands. For example, consider

∗Correspondence to: Ala Al-Fuqaha, Department of Computer Science, Western Michigan University, 1903 W. Michigan Ave., Kalamazoo, MI 49008, U.S.A.
†E-mail: alfuqaha@cs.wmich.edu

Copyright © 2009 John Wiley & Sons, Ltd.
mobile computing, the goal is simple, that is, to remain connected to the global internet untethered and still have the ability to perform the various tasks and operate applications as though connected to the internet traditionally (physically). The demand from mobile computing is simple: seamless ubiquitous connectivity. The research in that paradigm is continuously trying to meet this demand via research into optimal coverage schemes, connectivity speeds, mobile wireless internet standards, etc.

Similarly, consider pervasive computing, the goal, according to Sun MicroSystems [1], is to converge computers, communication, consumer electronics, content, and services, such that devices interact over two underlying layers, namely, service and standards [1]. The service layer establishes infrastructure for computing, communication, content, access, etc. whereas the standards layer allows information and application exchange (e.g., standards include, Java, XML, HTML, etc.) [1]. The demand from pervasive computing is simple: seamless ubiquitous computing/communication. The research in this paradigm is continuously trying to achieve the above-mentioned goals.

Let’s consider the evolving SOC, where services provide a higher level abstraction to traditional applications and users are considered as consumer of these services that lookup, via registries, providers/producers of these services. The goal again is to meet the demand for ubiquitous computing through a higher level of abstraction of applications as services.

Therefore, it can be seen that the current demand from this evolving global internetworking technology is the same—seamless, untethered, ubiquitous, autonomous computing and communication.

Meeting this demand is in essence a two-fold approach, first, developing a ubiquitous, autonomous environment, and second, developing a mechanism for mapping consumers to providers/producers. As mentioned earlier, numerous ubiquitous computing environments and various service advertisement, discovery and invocation protocols, such as, Jini, Salutation, etc. and standards, such as, XML, WSDL, OWL, etc. are continuously being explored and improved.

We contribute to this area of research by presenting a novel service location and planning (SLP) methodology that not only provides a mechanism for service invocation but also accounts for quality of service (QoS) parameters such as throughput and delay. This is a unique methodology since to the best of our knowledge, QoS requirements have not been factored into service discovery protocols and standards.

The SLP methodology can be used in various computing paradigms, ranging from application oriented network (AON) technology [2], to the newer computing paradigm of opportunistic networks [3], to more traditional paradigms of pervasive computing, SOC, and mobile computing.

We will present the SLP problem in terms of AON technology that enables the relocation of application services to adapt to changes in the underlying network conditions [2]. This is a huge impact technology since it allows applications to be relocated on the network in support of QoS.

This has two obvious fundamental consequences. First, AON nodes can route application-layer messages, such as stock quotes, weather and news alerts, etc., to the appropriate application service rather than an arbitrary IP address. Second, services can be installed on AON nodes. This enables faster satisfaction of service demand requests, as these requests are not propagated to the end-points of the network and instead are interpreted and satisfied by different nodes within the network.

Consider such an AON with \( n \) nodes in the network requesting \( s \) services with pre-determined throughout and delay parameters. The goal is to install services within the network to meet the requests with the QoS parameters, while minimizing service installation costs. In such scenarios, the SLP problem can be solved to solve the service installation problem optimally.

The SLP problem can be briefly stated as, given a network of nodes and a set of services that are being requested by nodes in the network. The problem associates a service installation cost with every service that is to be installed in the network to meet the requirements. These installation costs vary from one node to another. The goal is to minimize the service installation cost, promote service federation, and furthermore meet QoS requirements of throughput and delay. This is a novel approach, since to the best of our knowledge, SLP, and QoS requirements have not been previously factored into AON and service discovery protocols and standards.

We contribute with the definition, mathematical formulation of the SLP problem as an integer linear programming (ILP) problem, and its Lagrangean relaxation (LR), and achieve optimal results and near-optimal results for small and large-scale networks.

The rest of the paper is organized as follows. In Section 2, present an overview of other similar
optimization problems and compare the SLP problem with them to show the uniqueness of the SLP problem. In Section 3, we present an overview of approximation techniques for ILP problems and discuss their necessity for large-scale problems and present a small tutorial on the LR technique used in this paper in Section 4. In Section 5, we present a formal definition of the SLP problem and a detailed discussion on the mathematical formulation of the SLP problem as an ILP problem in Section 6. In Section 7, we discuss the LR of the SLP problem. In Section 8, we discuss test scenarios and results from ILP and LR of the SLP problem. We conclude with a review of our findings and contributions in Section 9.

2. Background—Comparison of Some Optimization Problems with SLP

In a survey of the various different optimization problems that are being researched, we have identified that the SLP problem can seem similar to the popular uncapacitated/capacitated facility location problems (UCFL) [4], however, a closer look at the SLP and UCFL will show that the SLP is broader and more complex than the NP-hard UCFL [5].

The generic facility location problem can be stated as given a set of potential facility locations $F$ and a set of sites to serve, say $C$, find a mapping that enables all sites to be served by opening a minimal number of facilities at the locations from the set $F$.

However, in SLP problems, the input is only the set of service requests, like the set of ‘sites to be served’ whereas the set of ‘location of facilities’ ($F$) to serve the requests is, in essence, the entire network. Thus, the facility location problem is a subset of the SLP problems, when we force the producers to belong to a subset of the nodes in the network.

Other problems of interest, such as optimal placement of gateways in wireless mesh networks [6,7] and service selection to minimize cost or maximize QoS [8], are also considered as variants of facility location problem, and have similar difference to that of SLP and facility location problems.

A review of networking and graph theory related optimization problems shows that vertex cover problems and its like seem related to SLP problems. However, a close look at vertex cover problems as a basis for SLP problem formulation and definition shows that it does not allow us to formulate QoS constraints in a vertex cover problem that can install services. For example, vertex cover in undirected graphs, such as in Reference [9] does not solve our SLP problem, and vertex cover in directed graphs [10] could be considered as a basis for heuristic and meta-heuristic approach to SLP problem but not for ILP formulation for optimal solutions.

3. Background—Approximation Techniques

Mathematical formulation of research problems as optimization problems is a research methodology that is used in diverse areas of research and study ranging from economics to physics to computer science. When a research problem can be successfully formulated as a mathematical problem/model, it can be expressed and defined unambiguously and solved accurately. However, often hard or complex formulations cannot be solved effectively or are computationally intensive. Furthermore, some mathematical formulations can only be solved on a small-scale and when large-scale scenarios are considered, the problem becomes too large or complex to be solved using traditional ILP/LP solve engines (e.g., CPLEX, LpSolve [11], etc.). In such cases, approximation techniques, such as linear programming (LP) relaxation and LR, are used to solve the problem. Historically, LR is similar to the Dantzig–Wolfe and benders decomposition [12].

With LP relaxation, the integer constraints of a linear program are ignored. However, generally the solution to a LP relaxed problem does not fulfill the discrete requirements [13].

In constrained optimization problems, LR prove particularly useful for separable nonlinear programming problems or for ILP problems [14]. Separable nonlinear programming problems are those that contain some linear parameters/constraints, and removing them from the problem leaves a problem involving only nonlinear parameters [15]. Whereas, ILP problems are those that consist of linear functions/constraints comprised of integer variables.

A useful observation of hard problem is that they can be viewed as an easy problem complicated by a relatively small set of side constraints [16]. By ‘dualizing’ the complicating constraints, a simpler, with reference to the original problem, easy to solve Lagrangean problem is obtained [16].

Thus, we are interested in LR of ILP problems, where multipliers, also known as Lagrangean multipliers, are
attached to the complicating constraints and moved into the objective function [15]. In this way, the multipliers are used to control the Lagrangean dual, such that, when a solution to the problem violates the complicating constraint, the multipliers are used to penalize the objective function. The optimal value of the Lagrangean dual objective function gives a lower bound for minimization problems and an upper bound for maximization problems, if multipliers are always positive.

Let’s briefly illustrate the LR technique. Consider for example, an ILP minimization problem \( L \) defined as follows:

\[
L = \min f x
\]

subject to

\[
Ax \leq b \tag{1}
\]

\[
Cx = d \tag{2}
\]

\[
x = \{0, 1\} \tag{3}
\]

where \( x \) is the problem variable and \( f, A, \) and \( C \) are problem dependent input parameters and \( b \) and \( d \) are constants. Note that, the third constraint makes this a combinatorial problem.

Let us assume that the first constraint is the complicating constraint.

A naive relaxation approach would be to remove the complicating constraint (1) and solve the problem without the complicating constraint. If the solution satisfies the complicating constraint then it is optimal for the original problem. However, using the LR for ILP problems, we formulate a Lagrangean dual \( L^* \) that is defined below and takes into account the complicating constraint(s) [17,18].

\[
L^* = \min (fx + \lambda_1 (Ax - b))
\]

subject to

\[
Cx = d \tag{2}
\]

\[
x = \{0, 1\} \tag{3}
\]

where \( \lambda_1 \) is the Lagrangean multiplier.

Therefore, for a valid solution for \( x \) and for \( \lambda_1 > 0 \)

\[
\Rightarrow Ax \leq b, \quad \text{since for valid solution } x, \text{ constraint (1) must be met, i.e. } Ax \leq b
\]

\[
\Rightarrow Ax - b \leq 0
\]

\[
\Rightarrow \lambda_1 (Ax - b) \leq 0, \quad \text{given } \lambda_1 = 0
\]

\[
\Rightarrow L^* = L, \quad \text{Note : } L^* = L \text{ is not guaranteed}
\]

This is how, the Lagrangean multipliers are used to control the solution \( x \) and penalize the objective function when a solution to \( x \) violates the complicating constraint(s).

Again, note that the Lagrange relaxation dual forms a lower bound for minimization problems, provided that the multipliers are not negative. In other words, we try to find the highest lower bound of the Lagrangean dual for close-to-optimal solution of the original problem. However, if the multipliers are allowed to be negative, then \( L^* > L \) or \( L^* \leq L \) could still hold true, in the case where the negative multipliers are small in magnitude. Thus, if the multipliers are not bounded to be positive the Lagrangean dual will not produce a bound for the problem, instead it will just converge to a valid solution irrespective of the cost.

Nonetheless, goal of the Lagrange relaxation technique is to iteratively update the Lagrange multipliers through different algorithms, such as subgradient, surrogate gradient [14,15], a technique we refer to as the CMU (Carnegie Mellon University) technique [19], etc. However, irrespective of the technique used, the multipliers are updated such that they represent the cost of violating the constraint it represents. Therefore, we only briefly review some of the LR techniques, namely, the subgradient, surrogate gradient, and the CMU techniques.

3.1. Subgradient and Surrogate Gradient Techniques

For simplicity, we omit the mathematical proofs and theorem and just discuss the intuition behind subgradient and surrogate gradient methods and refer readers to Reference [14] for a more detailed discussion.

The Lagrangean dual is nondifferentiable at an optimal point, but subdifferentiable everywhere else [16]. Thus, the subgradient method is an adaptation of the gradient method in which the gradient is substituted with the subgradient [16]. In this method, the multipliers are updated according to the step-size and subgradient of the dual Lagrangean function.
For a binary linear min. problem, with all constraints in canonical form
1. Begin with each $\lambda_i$ at 0, with step size $k$ (problem dependent value)
2. Solve the Lagrangean dual to get current solution $x$.
3. For every constraint violated by $x$, increase corresponding $\lambda_i$ by $k$.
4. For every constraint with positive slack relative to $x$, decrease the corresponding $\lambda_i$ by $k$.
5. If $m$ iterations have passed since the best relaxation value has increased, cut $k$ in half.
6. Go to 2.

Fig. 1. CMU Lagrangean relaxation technique [19] adapted for binary linear minimization problems.

at each iteration. The step-size is an application dependent variable and can be computed according to a mathematical formula or an easier rule that has performed well empirically is to set step size to 2 and halved whenever Lagrangean dual bound fails to increase after some fixed number of iterations [15].

Computing the subgradient requires a solution of all the sub-problems [14].

The surrogate gradient minimizes time by eliminating the need for a solution to all sub-problems, as in subgradient approach [15].

3.2. CMU Lagrangean Relaxation Technique

In the multiplier update method of Reference [19], which we termed CMU method, time consumption is shortened significantly as the need to compute any subgradient or surrogate gradients is eliminated and a very straightforward approach is used to update the multipliers. We adapted this algorithm for minimization problems and present it in Figure 1. In short, the multipliers are updated by a constant $k$ in each iteration based on whether they violate or give slack to a constraint.

Provided all these different techniques for implementing Lagrange relaxation, two properties are important in evaluating which relaxation technique to use, the sharpness of the bounds produced and the amount of computation time required to obtain these bounds [15]. Usually, there is a tradeoff between these two properties [15].

Our goal is simplicity, in terms of computationally intensity, thus we adapt the CMU LR technique to extend the SLP model for large-scale networks.

4. Example—CMU Lagrange Relaxation of a Small Binary Minimization Problem

In this section, we demonstrate the CMU LR technique that has been adapted for binary linear minimization problems (cf. Figure 1).

Consider the simple minimization problem defined below in its canonical form:

$$ Z = \min 4x_1 + 5x_2 + 6x_3 + 7x_4 $$

subject to

$$ 2x_1 + 2x_2 + 3x_3 + 4x_4 \leq 7 \quad (1) $$

$$ x_1 - x_2 + x_3 - x_4 \leq 0 \quad (2) $$

$$ -x_1 - x_2 \leq -2 \quad (3) $$

$$ x_j \in \{0, 1\}, \forall 1 \leq j \leq 4 $$

Now, given, we have step size $k = 0.5$ and $m = 5$ and $Z^*$ as above.
4.2. In Iteration 2

We have all $\lambda_i = 0$, $1 \leq i \leq 3$

$$\Rightarrow \min \begin{cases} 4x_1 + 5x_2 + 6x_3 + 7x_4 \\ +0 \left( 2x_1 + 2x_2 + 3x_3 + 4x_4 - 7 \right) \\ +0 \left( x_1 - x_2 + x_3 - x_4 - 0 \right) \\ +0 \left( -x_1 - x_2 + 2 \right) \end{cases}$$

$$= \min 4x_1 + 5x_2 + 6x_3 + 7x_4$$

Thus, the bound is 0, with all $x_j = 0$, $1 \leq j \leq 4$. This solution gives positive slack to constraint (1), just meets constraint (2), and violates constraint (3), thus the multipliers are updated and are $\lambda_1 = -0.5$, $\lambda_2 = 0$, and $\lambda_3 = 0.5$.

4.2. In Iteration 2

The Lagrangean dual becomes

$$\min \begin{cases} 4x_1 + 5x_2 + 6x_3 + 7x_4 \\ -0.5 \left( 2x_1 + 2x_2 + 3x_3 + 4x_4 - 7 \right) \\ +0.5 \left( -x_1 - x_2 + 2 \right) \end{cases}$$

$$= \min 2.5x_1 + 3.5x_2 + 4.5x_3 + 5x_4 + 4.5$$

The optimal solution to this dual is $x_j = 0$, $1 \leq j \leq 4$ with bound = 4.5 and multipliers are updated to $\lambda_1 = -1$, $\lambda_2 = 0$, and $\lambda_3 = 1$. Since the current solution to the LR dual gives positive slack to constraint (1), violates constraint (3), and still meets constraint (2).

In this manner, we play with the multipliers until iteration 14 when the bound, the value of the LR dual, has failed in five ($m = 5$) iterations. In this case, we cut the step size, $k$, of 0.5 in half, to get $k = 0.25$ and at which point the multipliers are $\lambda_1 = -2$, $\lambda_2 = -0.5$, and $\lambda_3 = 3$. We continue to update the multipliers as demonstrated in iterations 1 and 2, above.

In iteration 16, we find a higher bound of 14.25, and in iteration 21, when the bound fails to increase any further, we cut the step size in half, yielding the new step size of $k = 0.125$.

We continue in this manner to iteration 23, where the bound of 14.5 is achieved, and when the bound fails to increase in the next five iterations, in iteration 28, we cut the step size to 0.0625. In iteration 33, the bound does not increase any further and again we cut the step size to half, to get $k = 0.03125$.

In our program, our termination criterion is $k < 0.05$. Thus, we terminate the Lagrangean relaxation technique in 33 iterations. At which point, the solution is $x_1 = 1$, $x_2 = 1$, $x_3 = x_4 = 0$, and the bound = 14.5.

This solution set is the optimal solution set, and when substituted into the original objective function ($Z$) we get the optimal objective value of 9. The Lagrange multipliers at this termination stage are $\lambda_1 = -1.875$, $\lambda_2 = 0.4375$, and $\lambda_3 = 3$.

In this manner, we get the optimal solution of $x_1 = 1$, $x_2 = 1$, $x_3 = x_4 = 0$ with a bound = 14.5 (the LR dual value), which is higher than the optimal objective function value. In this case, the LR bound did not provide a lower bound of the original problem; this is because we allowed the multipliers to be negative.

Furthermore, scrutiny of this simple CMU technique yields the fact that this Lagrange relaxation technique only works for binary linear integer programming problems. Therefore, we will further adapt it for our SLP problem that is an ILP problem. However, before we do so, we will present and discuss the ILP formulation of the SLP problem.

5. Service Location and Planning (SLP) Problem—A Formal Definition

The SLP Problem can be defined as follows.

5.1. Given

A network graph $G = (V, E)$ and a set of services $S$, where $V$ is set of vertices/nodes and $E$ is the set of edges. There exists a set of consumers, $V^C \subseteq V$, such that all nodes $v \in V^C$ are requesting a service(s), $s_i \in S$, $i = 1, ..., |S|$ and have a throughput and delay demand associated with each request. There is also a service installation cost associated with each service $s$ on a node $n$ in the network $G$ and a service discount $\Gamma$ is given to promote service federation (multiple service installations on the same node).

5.2. Problem

Install services on a set of producers $V^P \subseteq V$, such that the service installation cost incurred is minimal and all throughput and delay requirements are satisfied, while also satisfying the underlying link layer capacities. Note: producers can also be consumers of services, $\Rightarrow V^P \cap V^C \neq \emptyset$.

6. ILP Formulation of SLP Problem

Here, we begin by discussing some assumption made and then delineating and defining the input, output, and variables of the ILP formulation.
6.1. Assumptions

Delay in a network is attributed to queuing delay at intermediate nodes, propagation delay and transmission delay. These can be formulated as

\[ \text{T}_{\text{queuing}} = \frac{1}{\text{link capacity} - \text{load on link}}, \]

\[ \text{T}_{\text{transmission}} = \frac{\text{data size}}{\text{link capacity}}, \]

and

\[ \text{T}_{\text{propagation}} = \frac{\text{link length}}{2\text{c}}, \]

where \( c \) is speed of light and the queuing delay is for \( \text{M/M/1 queues} \). Consequently, end-to-end delay formulation is nonlinear, due to the nonlinearity in end-to-end queuing delay. Note: accounting for transmission and propagation delay is trivial and not part of our formulation.

We use an approximation technique to formulate linear queuing delay constraints. The technique is simple. We use a load-delay lookup table to compute the nonlinear queuing delay at a link, by calculating the load on the link, and looking up the delay value for that load on that link in a load-delay lookup table. The load-delay lookup table is computed using a normalized \( T_{\text{queuing}} \) equation. In Figure 2, we illustrate a load-delay lookup tables for bandwidth capacities of 100, 300, 600, and 1000 Mb/s, this implies that as long as the maximum bandwidth capacity in the network being considered is 100, 300, 600, or 1000 Mb/s, we can approximate the queuing delay at the respective link by looking up the load on that link.

For example, for a link with bandwidth capacity 100 Mb/s and load 50, the queuing delay can be approximated as 200 s. In this case of the load-delay lookup table of Figure 2, the load lookup interval is 50. This implies, that if, a link with capacity 100 Mb/s had a load of 30, the delay would be undefined, since the table only delineates delay for loads of 0, 50, and 100 (for 100 Mb/s link). We round up the load on a link to the nearest lookup value, so that such cases are defined.

For example, for a link with bandwidth capacity of 600 Mb/s and load of 137, in the ILP formulation we round up the load so that the delay on this link is equivalent to the delay with load of 150, that is 22 s.

Therefore, there is a tradeoff between bandwidth wastage and lookup table storage. If the lookup table has unit increments of load then the lookup table will be huge to allow lookup for all loads ranging from 0, 1, 2, ..., \( \text{max\_bandwidth} \), since \( \text{max\_bandwidth} \) also caps maximum load on a link in the network. On the other hand, if lookup table is in increments of, say 100, then we are wasting bandwidth since delay for load of 225 will be undefined and load will have to be rounded up to 300 for a load-delay lookup in the table.

However, to overcome the nonlinearity in delay formulations we compromise with bandwidth wastage. Some of the load-delay lookup tables used in our implementation are illustrated in Figure 2, with load interval of 50 in the delay lookup table.

Consequently, we realize that bandwidth resources are wasted, if a link with load = 225 is rounded up and delay for load 300 is assigned to this link. However, this is a tradeoff we make to formulate linear delay constraints. Furthermore, the bandwidth wasted is directly proportional to the scale used in the lookup table, which in turn is directly proportional to the storage required for the lookup table.

A second assumption we make to allow local service installation is that all nodes have a zero-cycle loop with \( \text{max\_bandwidth} \) of 1000 Mb/s. In this case, when needed services can be stored locally. This implies that there will be a delay incurred in accessing locally...
installed services, this delay can be assumed to be processing overhead, which will be very small since the bandwidth capacity can be very high, e.g. 1000 Mb/s in our simulations.

6.2. Input

The inputs of the problem can be delineated as

1. \( n \) = number of nodes in the network, i.e. \(|\mathcal{V}|\).
2. \( s \) = maximum number of services requested.
3. \( p \) = maximum number of useful paths in the network.
4. \( e \) = total number of links/edges in the network, i.e. \(|\mathcal{E}|\).
5. Path-link matrix \( L \) is a binary \( p \times e \) dimensional that indicates whether a link is used in a path or not.
6. Bandwidth vector \( B \) is \( e \) dimensional that gives capacity of a link/edge and \( b_{\text{max}} \) is maximum link capacity (in the entire network), where \( b_l \) is the bandwidth of link \( l \).
7. Service installation cost matrix \( C \), is \( n \times s \) dimensional that quantifies the cost for installing a service on a node.
8. Discount \( \Gamma \) is given if multiple services are installed on node.
9. A three dimensional binary routing matrix, \( R \) that indicates the path used between a source–destination pair in the network.
10. A throughput demand matrix \( T \) that gives the throughput required for a service at a consumer.
11. A delay demand matrix \( D \) that is the maximum delay allowed for a service at a consumer.

6.3. Variables and Their Definitions

The variables of the problem can be defined as follows:

1. A binary service location matrix \( X \), is \( n \times s \) dimensional, that indicates whether a service is installed on a node or not.
2. A path-service capacity matrix, \( Z \) is \( p \times s \) dimensional, that quantifies the capacity of a service on a path.
3. An \( n \) dimensional service installation indicator vector \( U \) that indicates whether multiple services are installed on a node.
4. A binary service-path indicator (normalized \( Z \)) matrix \( Y \) is \( p \times s \), and indicates if a service uses a path or not.
5. An \( e \) dimensional link load vector \( V \) that quantifies the load on a link.
6. An \( e \) dimensional link-delay vector \( G \) that gives the delay on a link.
7. A path delay vector \( H \), gives the total end-to-end delay on a path.
8. An indicator variable \( r \) is \( n \times n \times s \times p \), such that \( r_{i,j,k,m} = \begin{cases} 1, & \text{if } i \text{ provides service } k \text{ to } j \text{ via path } m \\ 0, & \text{otherwise} \end{cases} \)

Variable \( X \) is also the output of the problem since it gives the optimal service installation configuration.

6.4. ILP Formulation

Based on the inputs and variables, we formulate the ILP model as follows (also in previous work [20]).

6.4.1. Minimize the cost of service installation

\[
\min \left\{ \sum_{i=1}^{n} \sum_{k=1}^{s} C_{i,k} \cdot X_{i,k} + \sum_{i=1}^{n} \Gamma \cdot U_i \right\}
\]

subject to

6.4.2. Service installation cost and location constraints

\[
\sum_{k=1}^{s} (X_{i,k}) - p \cdot b_{\text{max}} \cdot U_i \leq 0, \quad \forall 1 \leq i \leq n
\]
\[
-\sum_{k=1}^{s} (X_{i,k}) + U_i \leq 0, \quad \forall 1 \leq i \leq n
\]

6.4.3. Throughput constraints

\[
-\sum_{i=1}^{n} \sum_{m=1}^{p} R_{i,j,m} \cdot Z_{m,k} \leq -T_{i,k}, \quad \forall 1 \leq i \leq n,
\]
\[
1 \leq k \leq s
\]
6.4.5. Delay constraints

\[ X_{i,k} - \sum_{j=1}^{n} \sum_{m=1}^{p} R_{i,j,m} \cdot Z_{m,k} \leq 0, \quad \forall 1 \leq i \leq n, \quad 1 \leq k \leq s \]  
\[ -p \cdot b_{\text{max}} \cdot X_{i,k} + \sum_{j=1}^{n} \sum_{m=1}^{p} R_{i,j,m} \cdot Z_{m,k} \leq 0, \quad \forall 1 \leq i \leq n, \quad 1 \leq k \leq s \]  
(4)

6.4.7. Finding index to be looked up in the delay table

\[ V_l - q \cdot i - (b_{\text{max}} + q_{\text{max}}) \cdot c_{l,i} \leq 0, \quad \forall 1 \leq l \leq e, \quad 1 \leq i \leq \frac{b_l}{q} + 1 \]  
\[ -V_l + q \cdot i - (b_{\text{max}} + q_{\text{max}}) \cdot c_{l,i} \leq 0, \quad \forall 1 \leq l \leq e, \quad 1 \leq i \leq \frac{b_l}{q} + 1 \]  
(13)
6.4.10. To make sure only those paths offer a service capacity that are between a producer and a consumer

\[-q_{\text{max}} \cdot p \sum_{i=1}^{n} \sum_{m=1}^{p} R_{i,j}^{m} Y_{m,k} \leq -T_{j,k}, \quad \forall 1 \leq j \leq n,\]

\[1 \leq k \leq s\]  

(25)

\[\sum_{i=1}^{n} \sum_{m=1}^{p} R_{i,j}^{m} Y_{m,k} \leq T_{j,k}, \quad \forall 1 \leq j \leq n, 1 \leq k \leq s\]

(26)

7. ILP Formulation Discussion

Here, we interpret the constraints presented in the above mathematical formulations. The constraints relate to: service installation and location, meeting throughput, factoring underlying network link layer capacity, ‘discretization’ of load, approximating delay for load by performing a lookup in a load-delay table, computing end-to-end delay on a path and meeting delay requirements.

In the service location problem, our goal is to install service(s) on nodes in the network that meet throughput and delay requirements for requests, while minimizing the service installation costs and promoting service federation. Therefore, the objective function is to minimize the service installation cost and give a discount for multiple service installations, captured by the service installation indicator vector U. We capture service installation location details in constraints (1) and (2) where \(X_{i,k} = \begin{cases} 1, & \text{if service } k \text{ is installed on } i \\ 0, & \text{otherwise} \end{cases}\).

Constraint (3) ensures that the sum of service capacity provided across all paths leading to destination \(j\) (consumer \(j\)) with throughput demand of \(T_{j,k}\) for service \(k\) is met. Constraints (4) and (5) ensure that the paths providing the service capacity come from nodes where the service is installed, that is from a service provider \(i\).

Constraint (6) captures the network link layer capacity and restricts the service capacity on a path so that it does not exceed the underlying individual link capacity of the links in a path. Constraints (7) and (8) compute the service-path indicator, a binary variable \(Y_{m,k} = \begin{cases} 1, & \text{if } Z_{m,k} \geq 1 \\ 0, & \text{otherwise} \end{cases}\).

The first six constraints allow us to meet throughput requirements while minimizing service installation costs. However, our SLP problem meets QoS requirements of throughput and delay. Therefore, we next discuss the constraints to meet the delay requirements.

Constraints (9) and (10) compute the load on a link as the sum of service capacity across all paths using the link and are equivalent to \(V_{l} = \sum_{m=1}^{p} \sum_{k=1}^{s} (L_{m,l} \cdot Z_{m,k})\). Constraint (11) ensures that load on the link, \(V_{l}\), is defined in the load-delay lookup table \(Q\) with load interval \(q\), with \(a_{l} = V_{l}/q\) (equivalent to constraints (11) and (12)). Constraints (13) and (14) check whether there is a difference between load on a link and the respective lookup load in the table for a given link. Constraint (15) ensures that there is at least one entry in the lookup table that matches the load on the link, indicated by 0. Constraints (16) and (17) compute \(1 - c_{l,i}\), so that the lookup can be performed, which will be achieved by multiplying \(k_{l,i}\) with \(Q_{l}\) in the respective row that matches the maximum bandwidth of the link. Constraints (18) and (19) compute the delay on a link \(G_{l}\) as \(G_{l} = \sum_{i=1}^{q} (k_{l,i} \cdot Q_{l})\).

End-to-end delay of a path is computed as the sum of the delay across all links in the path, that is, \(H_{m} = \sum_{l=1}^{p} (L_{m,l} \cdot G_{l})\) (equivalent to constraints (20) and (21)). Constraints (22–24) ensure that the end-to-end delay of a path does not exceed the delay requirement \(D_{j,k}\). That is, every path between a service provider \(i\) and consumer \(j\) for service \(k\) must not exceed delay requirement of consumer for service \(k\), equivalent to \(R_{i,j,m}^{n} Y_{m,k} \cdot H_{m} \leq D_{j,k}\). However, this is nonlinear, therefore, constraints (22) and (23) compute a binary variable \(r_{i,j,k,m} = \begin{cases} 1, & \text{if } R_{i,j,m}^{n} Y_{m,k} = 0 \\ 0, & \text{otherwise} \end{cases}\), so if \(r_{i,j,k,m} = 1\), then the path delay and delay requirement comparison is not valid, however, if \(r_{i,j,k,m} = 0\), then we want to ensure that the path delay does not exceed delay requirement, this is captured in constraint (24). Two safety constraints, constraints (25) and (26) are added to ensure that only those paths carry services or offer services that lead from a provider to a consumer.

8. Lagrangean Relaxation of SLP Problem

As discussed earlier, ILP models can be solved for small scale networks efficiently, however, for larger
scale networks relaxations techniques, approximation techniques, and other heuristics and meta heuristics have to be used, which may yield suboptimal results, but allowing solution for large scale networks, that cannot be handled by LP solve engines or solvers in short.

Recall, Lagrange relaxation techniques are used to remove/relax complicating constraints from the ILP model, making the model simpler, and associating a multiplier with the relaxed constraint (the removed constraint) so that cost of violating or removing the constraint is captured by the multiplier.

Recall, we wanted to use the simple and straightforward approach of the CMU technique. We adapted the CMU technique for binary linear minimization problems in Figure 1. However, the technique is only applicable to binary linear minimization problems, and our SLP problem is an integer linear minimization problem. Therefore, we only relax/remove those constraints that are constituted of binary variables, such as constraints (1), (2), and (17) in the ILP formulation of Section 6. We will show significant reduction in the number of constraints in the Lagrangean dual of the SLP problem even with just relaxing the above-mentioned three constraints.

9. Test scenarios and results From ILP and Lagrangean Relaxation of SLP

All our test scenarios are based on the topology of a carrier’s nation-wide IP backbone network topology that is illustrated in Figure 3. For small-scale networks, we abstract smaller topologies from this larger scale network. For example, a six-node network can be abstracted with nodes 1, 2, 3, 4, 7, and 6. In Figure 4, we use this small-scale network to illustrate how SLP optimally installs services in the network to meet service requests and QoS parameters. This ILP formulation of the SLP problem was solved using the LpSolve engine [11].

In the scenario illustrated in Figure 4, it is optimal to install service 1 on node 3 and service 2 on node 5. Furthermore, service splitting occurs since all paths between nodes 3 and 1 are bounded by the bandwidth capacity of the lowest link, i.e. 100 Mb/s. So, 100 Mb/s of service 1 are provided via the direct path 3,1 and the indirect path 3,2,1 provides 50 Mb/s of service 1. We exceed node 1’s throughput requirement of 137 by 13 Mb/s, since the delay lookup table in Figure 2 was used, which is enumerated for loads of increments of 50 units. Thus, instead of providing 37 Mb/s on the indirect path, 50 Mb/s are provided. It is true that some bandwidth is wasted however the tradeoff is linear computation of delay.

We also meet delay requirements, since each path from node 3 delivering service 1 to node 1, does not exceed the delay requirement of 1000 s. The direct path 3,1 has a total end-to-end delay of 500 s (from the load-delay lookup table in Figure 2). The indirect path 3,2,1 has a total end-to-end delay of 400, 200 s delay on link 3,2, and 200 s delay on link 2,1 (from the load-delay lookup table in Figure 2).

Similarly, it is optimal to install service 2 on node 5. In this way, we incur minimal service installation cost and meet delay and exceed throughput requirements of node 6. We deliver 200 Mb/s of service 2 (same reason as above to exceed throughput) via the direct path 5,6 with a delay of 16 s (using load-delay lookup table like the one in Figure 2 for link with bandwidth capacity

Fig. 3. A carrier’s nation-wide IP backbone network topology [21].
800 Mb/s), less than the QoS delay requirement of 1000 s.

For node 3’s request of 850 Mb/s of service 3 with a delay of no more than 100 s. Firstly, note service 3 is cheapest on node 3 and secondly, with service federation we will get a discount of 20 off the total service installation cost. Thirdly, due to the low delay requirement, we are forced to install service 3 on node 3, since we cannot meet the delay requirement with service installation on any other node, e.g. just a load of 50 on link with bandwidth 100 Mb/s is 200 s delay. Thus, service 3 is installed on node 3 and 850 Mb/s of service 3 are provided to node 3 locally, through the zero-cycle loops with 1000 Mb/s, with a delay of 67 s (from Figure 2).

In this manner the ILP formulation achieves optimal service installation costs for small-scale networks. However, for the SLP problem discussed in this paper, it is obvious that there are various levels of complexity. The dimensions of complexity in a SLP can be delineated in terms of

- number of nodes, $n$,
- number of services, $s$,
- number of paths, $p$,
- links/edges, $e$,
- bandwidth capacity, $b_l$ (cf. ILP formulation in Section 6),
- and lookup table interval, $q$.

Increasing any of these dimensions causes the problem to grow tremendously. For example, consider a problem with 6 nodes, 2 services, and 22 paths, the number of variables in this problem is 480, where as, a 6 node, 3 service, and 22 path problem has 538 variables, leaving all other dimensions to be the same across the two problems, this is an increase of 58 variables just with respect to an increase of one service in the SLP problem.

Thus, we use the LR technique for extending the ILP formulation of the SLP problem for large-scale networks. Using the LR technique (discussed in Section 7), we compare the reduction in the number of constraints in a problem in Figure 3 (also tabulated in Table 1). This illustrates the improvement with LR over the ILP formulation of the SLP problem.

Figure 5 illustrates a reduction of 15–18% (at the least 15.6% and at the most 18.8%) of the total constraints, i.e. an average improvement of 17.42% (cf. Table 1). This yields reductions of up to 729 constraints in a 24 node, 2 service and 58 path network setup. We reduce the complexity of the problem by only utilizing
Table I. ILP cost and LR lower bound comparison.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>ILP cost</th>
<th>LR lower bound</th>
<th>Constr.(^a) in ILP</th>
<th>Constr.(^a) in LR</th>
<th>Improvement in constr.(^a) (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6n-2s-22p</td>
<td>240</td>
<td>219.62</td>
<td>940</td>
<td>778</td>
<td>17.23</td>
</tr>
<tr>
<td>6n-3s-22p</td>
<td>340</td>
<td>324.93</td>
<td>1038</td>
<td>876</td>
<td>15.61</td>
</tr>
<tr>
<td>6n-3s-22p</td>
<td>340</td>
<td>298</td>
<td>1158</td>
<td>966</td>
<td>16.58</td>
</tr>
<tr>
<td>10n-2s-26p</td>
<td>240</td>
<td>220</td>
<td>1528</td>
<td>1250</td>
<td>18.19</td>
</tr>
<tr>
<td>10n-2s-26p</td>
<td>180</td>
<td>168</td>
<td>1656</td>
<td>1346</td>
<td>18.72</td>
</tr>
<tr>
<td>15n-2s-66p</td>
<td>60</td>
<td>48</td>
<td>2395</td>
<td>2001</td>
<td>16.45</td>
</tr>
<tr>
<td>20n-3s-82p</td>
<td>100</td>
<td>68</td>
<td>3594</td>
<td>3050</td>
<td>15.14</td>
</tr>
<tr>
<td>24n-2s-58p</td>
<td>160</td>
<td>236</td>
<td>3810</td>
<td>3095</td>
<td>18.77</td>
</tr>
<tr>
<td>24n-2s-58p</td>
<td>280</td>
<td>320</td>
<td>3872</td>
<td>3143</td>
<td>18.83</td>
</tr>
<tr>
<td>24n-2s-32p</td>
<td>160</td>
<td>220</td>
<td>3819</td>
<td>3104</td>
<td>18.72</td>
</tr>
</tbody>
</table>

Average improvement = 17.42

LR = Lagrangean relaxation.

\(^a\)Constr. = number of constraints.

**Fig. 5.** Comparison of number of constraints with ILP versus LR of SLP problem.

useful alternate paths (with a minimum number of edges traversed), rather than all possible paths between all source and destination pairs, which in itself is a hard problem.

The scenarios delineated in Figure 5 as \(x\)-\(y\)-\(z\)-\(p\), where \(x\), \(y\), and \(z\) are the number of nodes, services, and paths, respectively. If two scenarios are labeled the same, then the difference is the bandwidth capacity of an underlying link(s). We keep the number of edges \(e\) constant across scenarios with same \(n\), \(s\), \(p\) dimensions. Also, the same lookup table is used in all scenarios.

Table I shows how LR bounds the ILP objective function value. It should be noted that the Lagrangean dual does not always represent a lower bound on the optimal solution (ILP cost) for all scenarios since we allowed multipliers to be negative.

**10. Conclusion**

In conclusion, we illustrated a novel SLP problem that is unique in the way it relates consumers to producers while taking into account individual throughput and delay requirements and not exceeding underlying network link layer capacities. The aim of the SLP problem is to minimize service installation costs (promote service federation) while meeting consumer defined QoS constraints.

This SLP problem formulation and solution technique can be used in various different computing paradigms ranging from AON to mobile computing.

Our contribution lies in the formal definition of the SLP problem and a mathematical formulation of the SLP problem as an ILP problem that models the SLP problem without ambiguity and solves the SLP problem optimally for small-scale networks.

We extended these results to large-scale networks, by applying LR technique to the ILP formulation of the SLP problem. We presented and discussed LR and some techniques used to implement LR for ILP problems and illustrated an adaptation of a very simple and straightforward technique for LR of our SLP problem. Our contribution is the use of the LR approximation approach to perform near optimal SLP in large-scale networks in terms of number of nodes and offered and requested services.

In this paper, we achieved an improvement of 17.42\%, on average, with LR of the SLP problem, that is, a reduction of, on average, 17.42\% in the number of constraints in the SLP problem with LR.

Future work includes further reducing the number of constraints by re-modeling the ILP formulation of the SLP problem in terms of binary variables, and comparing results from different LR techniques in terms of accuracy, complexity in implementation, and execution time.
Our test scenarios for large-scale networks are based on a carrier’s nation-wide IP network topology. These results validate use of SLP for large-scale real-world networks. However, we continue to work on designing, developing and implementing genetic algorithms and network simulations using the SLP mechanism to identify its real-word uses and limitations.

Acknowledgements

The authors acknowledge Computational Science Center and Information Technology and Image Analysis (ITIA) Center. Any opinions, findings, conclusions, or recommendation expressed in the paper are those of the authors and do not necessarily reflect the views of the funding agencies or institutions.

References


Authors’ Biographies

Zill-E-Huma Kamal is an Adjunct Professor of Computer Science at Colorado Technical University. She received her B.Sc. in Computer Science from Western Michigan University, in 2002 and continued directly to pursue her Ph.D. She received her Ph.D. in Computer Science in April, 2008. Dr. Kamal has published technical papers, journals, and book chapters and reviewed papers in the areas of service location and resource utilization in wireless networks and power consumption analysis in sensor networks. She has taught undergraduate level programming courses and graduate level courses in Modern Computer Architecture, Modern Computer Design, Systems Engineering, and Systems Integration and Testing. Dr. Kamal has also given lectures in Design and Analysis of Algorithms, Computer Security and Theory of Automata/Computation. She has received teaching and research awards at department and university levels since 2004–2008. Her research interest include analysis and design of parallel and distributed algorithms, Mobile Computing, Wireless Sensor Networks, Opportunistic Networks, Pervasive Computing, resource utilization with QoS, Optimization, and Operational Research.
Ala Al-Fuqaha (S’00-M’04) received his M.S. and Ph.D. degrees in Electrical and Computer Engineering from the University of Missouri, in 1999 and 2004, respectively. Currently, he is an Assistant Professor at the Computer Science Department of Western Michigan University. Before joining Western Michigan University, Dr. Al-Fuqaha was a senior member of technical staff at LAMBDA Optical Systems in Reston, Virginia where he worked on the design and development of embedded routing protocols and network management systems. Prior to LAMBDA, Dr. Al-Fuqaha was a Software Engineer with Sprint Telecommunications Corp. where he worked as part of the architecture team. His research interests include intelligent network management and planning, QoS routing and performance analysis, and evaluation of high-speed computer and telecommunication networks. Dr. Al-Fuqaha has served as Technical Program Committee member of many IEEE conferences.

Ajay Gupta is a Professor of Computer Science at Western Michigan University. From 1998 to 2002, he was the Chairman of the Computer Science Department at Western Michigan University. Dr. Gupta received his Ph.D. in Computer Science from the Purdue University in 1989, his M.S. in Mathematics and Statistics from the University of Cincinnati in 1984, and his B.E. (Honors) in Electrical and Electronics Engineering from the Birla Institute of Technology and Sciences, Pilani, India in 1982. Dr. Gupta’s research interests include sensor systems, mobile computing, web technologies, computer networks, evolutionary computation, scientific computing, and design and analysis of parallel and distributed algorithms. He has published numerous technical papers and book chapters in refereed conferences and journals in these areas. A paper he co-authored, ‘Adaptive Integration Using Evolutionary Strategies’, won the Best Paper award in the International Conference on High Performance Computing in 1996. Another paper he co-authored, ‘Lightweight Intrusion Detection for Sensor Networks’, received Honorable mention at the CERIAS Information Security Symposium in 2006. He also holds joint copyright for the parallel and distributed automatic numerical integration software package, ParInt 1.1. Dr. Gupta is a senior member of the IEEE and member of the IEEE Computer Society, the IEEE Communications Society, the ASEE, and the ACM. He actively helps organize various ACM and IEEE conferences.