Lecture Outline

- Example: Distributed spectrum management in DSL
- Distributed algorithm: introduction
- Primal and dual decomposition
- Gauss-Siedel and Jacobi algorithms
Example: Spectrum Management in DSL

Crosstalk phenomena similar to interference-limited wireless networks:

\[ R_i = \sum_{j=1}^{J} \log(1 + SIR_{ij}) \]

\( i \): user. \( j \): DMT tone

Key differences:

- Channel gains not time-varying
- Frequency selective
- Spatial dependence

Rate maximization for user \( i \):

\[
\begin{align*}
\text{maximize} & \quad R_i \\
\text{subject to} & \quad \sum_j P_{ij} \leq P_{i,\text{max}}, \\
& \quad P_{ij} \geq 0
\end{align*}
\]
Iterative Water-filling

Simultaneous update: Each user $i$ has a target rate:

- Allocate power by water-filling over interference plus noise spectrum for a given total power (iterative margin-adaptive water-filling)
- Change total power level based on attained rate (converges if target rates are feasible)

Proof of convergence to Nash equilibrium under certain conditions for two-user case

Nash equilibrium may not be optimal

Simple method to improve performance compared to static spectrum management
Distributed Algorithms

We have seen three distributed algorithms:

- **Shortest path routing**: Bellman Ford algorithm
- **Power control** in wireless and DSL

We will see more:

- **Network utility maximization**
- Rate allocation and **TCP congestion control**
- **Wireless NUM**
- **NUM extensions**
Distributed algorithms are preferred because:

- It’s scalable
- It’s robust
- Centralized command is not feasible or is too costly

Key issues:

- **Local computation** vs. **global communication**
- Scope, scale, and physical meaning of communication **overhead**
- **Theoretical issues**: Convergence? Optimality? Speed?
- **Practical issues**: Robustness? Synchronization? Complexity? Stability?
- Problem **separability structure** for decomposition: **vertical and horizontal**
**Decomposition: LP Example**

LP with variables $u, v$:

maximize $c_1^T u + c_2^T v$

subject to

$A_1 u \leq b_1$

$A_2 v \leq b_2$

$F_1 u + F_2 v \leq h$

**Coupling constraint:** $F_1 u + F_2 v \leq h$. Otherwise, separable into two LP
Primal Decomposition

Introduce variable $z$ and rewrite coupling constraint as

\[ F_1u \leq z, \quad F_2v \leq h - z \]

LP decomposed into a master problem and two subproblems:

\[
\text{minimize}_z \phi_1(z) + \phi_2(z)
\]

where

\[
\phi_1(z) = \inf_u \{ c_1^T u | A_1 u \leq b_1, F_1 u \leq z \}
\]

\[
\phi_2(z) = \inf_v \{ c_2^T v | A_2 v \leq b_2, F_2 v \leq h - z \}
\]

Subgradient of function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ at $x$ is a vector $g$ such that

\[
f(y) \geq f(x) + g^T (y - x), \quad \forall y
\]
**Primal Decomposition**

For each iteration $t$:

1. **Solve two separate LPs** to obtain optimal $u(t), v(t)$ and associated dual variables $\lambda_1(t), \lambda_2(t)$

2. **Subgradient update**: $g(t) = -\lambda_1(t) + \lambda_2(t)$

3. **Mater algorithm update**: $z(t+1) = z(t) - \alpha(t)g(t)$ where $\alpha(t) \geq 0$, $\lim_{t \to \infty} \alpha_t = 0$ and $\sum_{t=1}^{\infty} \alpha(t) = \infty$

**Interpretation:**

- $z$ fixes allocation of resources between two subproblems and master problem iteratively finds best allocation of resources
- More of each resource is allocated to the subproblem with larger Lagrange multiplier at each step
Dual Decomposition

Form partial Lagrangian:

\[ L(u, v, \lambda) = c^T u + x_2^T v + \lambda^T (F_1 u + F_2 v - h) \]
\[ = (F_1^T \lambda + c_1)^T u + (F_2^T + c_2)^T v - \lambda^T h \]

Dual function:

\[ q(\lambda) = \inf_{u,v} \{ L(u, v, \lambda) | A_1 u \leq b_1, A_2 v \leq b_2 \} \]
\[ = -\lambda^T h + \inf_{u: A_1 u \leq b_1} (F_1^T \lambda + c_1)^T u + \inf_{v: A_2 v \leq b_2} (F_2^T \lambda + c_2)^T v \]

Dual problem:

\[ \text{maximize} \quad q(\lambda) \]
\[ \text{subject to} \quad \lambda \geq 0 \]
**Dual Decomposition**

Solve the following LP in $u$, with minimizer $u^*(\lambda(t))$

\[
\begin{align*}
\text{minimize} & \quad (F_1^T \lambda(t) + c_1)^T u \\
\text{subject to} & \quad A_1 u \preceq b_1
\end{align*}
\]

Solve the following LP in $v$, with minimizer $v^*(\lambda(t))$

\[
\begin{align*}
\text{minimize} & \quad (F_2^T \lambda(t) + c_2)^T v \\
\text{subject to} & \quad A_2 v \preceq b_2
\end{align*}
\]

Use the following subgradient (to $-q$) to update $\lambda$:

\[
g(t) = -F_1 u^*(\lambda(t)) - F_2 v^*(\lambda(t)) + h, \quad \lambda(t + 1) = \lambda(t) - \alpha(t) g(t)
\]

**Interpretation:**

Master algorithm adjusts prices $\lambda$, which regulates the separate solutions of two subproblems
Parallelization

Parallelization of iterative algorithm: $x(t + 1) = F(x(t))$

- Gauss-Siedel algorithm
- Jacobi algorithm

Optimization problem with separable strictly convex objective:
- Cartesian product constraint set: distributed gradient algorithm
- Coupled constraint set: primal or dual decomposition

Optimization problem with separable convex objective:
- Proximal minimization algorithm
- Augmented Lagrangian method
- Distributed subgradient method
Jacobi and Gauss-Siedel Algorithms

In general, Jacobi algorithm ($F_i$ is $i$th component of function $F$):

$$x_i(t + 1) = F_i(x_1(t), \ldots, x_n(t))$$

Gauss-Siedel algorithm:

$$x_i(t + 1) = F_i(x_1(t + 1), \ldots, x_{i-1}(t + 1), x_i(t), \ldots, x_n(t))$$

Nonlinear minimization: Jacobi algorithm:

$$x_i(t + 1) = \arg\min_{x_i} f(x_1(t), \ldots, x_n(t))$$

Gauss-Siedel algorithm:

$$x_i(t + 1) = \arg\min_{x_i} f(x_1(t + 1), \ldots, x_{i-1}(t + 1), x_i(t), \ldots, x_n(t))$$

If $f$ is convex, bounded below, differentiable, and strictly convex for each $x_i$, then Gauss-Siedel algorithm converges to a minimizer of $f$.
Lecture Summary

- Decouple a coupling constraint: primal or dual decomposition
- Decomposition of optimization problems into subproblems for parallel algorithms: Jacobi or Gauss-Siedel algorithm