Parallel and Distributed Graph Cuts by Dual Decomposition

Petter Strandmark and Fredrik Kahl

Presented by Varun Gulshan

06 July, 2010
What is Dual Decomposition?

It is a technique for optimization: usually used for approximate optimization.

Dual: There is dual involved somewhere.
Decomposition: Some kind of decomposition involved.
What is Dual Decomposition?

- It is a technique for optimization: usually used for approximate optimization.
What is Dual Decomposition?

- It is a technique for optimization: usually used for approximate optimization.
- Dual: There is dual involved somewhere.
What is Dual Decomposition?

- It is a technique for optimization: usually used for approximate optimization.
- Dual: There is dual involved somewhere.
- Decomposition: Some kind of decomposition involved.
Consider an optimization problem of the form:

$$\min_y f_1(y) + f_2(y)$$
Consider an optimization problem of the form:

$$\min_y f_1(y) + f_2(y)$$

- Usually $f(y) = f_1(y) + f_2(y)$ is hard to optimize.
Consider an optimization problem of the form:

\[
\min_y f_1(y) + f_2(y)
\]

- Usually \( f(y) = f_1(y) + f_2(y) \) is hard to optimize.
- Each of \( f_1(y) \) and \( f_2(y) \) can be easily optimized separately.
Consider an optimization problem of the form:

$$\min_y f_1(y) + f_2(y)$$

- Usually $f(y) = f_1(y) + f_2(y)$ is hard to optimize.
- Each of $f_1(y)$ and $f_2(y)$ can be easily optimized separately.
- Dual decomposition idea: Decompose original problem into optimizable subproblems and combine their solution in a principled way.
Optimization Problem

Consider an optimization problem of the form:

$$\min_y f_1(y) + f_2(y)$$

- Usually $f(y) = f_1(y) + f_2(y)$ is hard to optimize.
- Each of $f_1(y)$ and $f_2(y)$ can be easily optimized separately.
- Dual decomposition idea: Decompose original problem into optimizable subproblems and combine their solution in a principled way.
- Use variable duplication and duality to achieve the above.
Original problem:

\[
\min_y f_1(y) + f_2(y)
\]
Optimization Problem

Original problem:

\[
\min_y f_1(y) + f_2(y)
\]

Duplicate variables:

\[
\min_{y_1, y_2} f_1(y_1) + f_2(y_2)
\]

\[
s.t \ y_1 = y_2
\]
Duplicate variables:

\[
\begin{align*}
\min_{y_1, y_2} & \quad f_1(y_1) + f_2(y_2) \\
\text{s.t} & \quad y_1 - y_2 = 0
\end{align*}
\]
Duplicate variables:

\[
\min_{y_1, y_2} f_1(y_1) + f_2(y_2)
\]

\[s.t \; y_1 - y_2 = 0\]

Write the lagrangian dual:

\[
g(\lambda) = \min_{y_1, y_2} f_1(y_1) + f_2(y_2) + \lambda^T (y_1 - y_2)
\]
Optimization Problem

The lagrangian dual:

\[ g(\lambda) = \min_{y_1, y_2} f_1(y_1) + f_2(y_2) + \lambda^T(y_1 - y_2) \]
The lagrangian dual:

\[ g(\lambda) = \min_{y_1, y_2} f_1(y_1) + f_2(y_2) + \lambda^T(y_1 - y_2) \]

Decompose the lagrangian dual:

\[ g(\lambda) = \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) + \min_{y_2} \left( f_2(y_2) - \lambda^T y_2 \right) \]
\[ = g_1(\lambda) + g_2(\lambda) \]
The lagrangian dual:

\[ g(\lambda) = \min_{y_1, y_2} f_1(y_1) + f_2(y_2) + \lambda^T (y_1 - y_2) \]

Decompose the lagrangian dual:

\[ g(\lambda) = \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) + \min_{y_2} \left( f_2(y_2) - \lambda^T y_2 \right) \]

\[ = g_1(\lambda) + g_2(\lambda) \]

where:

\[ g_1(\lambda) = \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) \]

\[ g_2(\lambda) = \min_{y_2} \left( f_2(y_2) - \lambda^T y_2 \right) \]
∀λ,  \( g(\lambda) \) is a lower bound on the optimal of the primal. So maximize the lower bound:

\[
\max_{\lambda} g(\lambda) = \max_{\lambda} \left( g_1(\lambda) + g_2(\lambda) \right)
\]
\( \forall \lambda, \ g(\lambda) \) is a lower bound on the optimal of the primal. So maximize the lower bound:

\[
\max_{\lambda} g(\lambda) = \max_{\lambda} \left( g_1(\lambda) + g_2(\lambda) \right)
\]

- The lagrangian dual \( g(\lambda) \) is always a concave function of \( \lambda \), so \(-g(\lambda)\) is convex, and hence is minimized easily.
∀λ, \( g(\lambda) \) is a lower bound on the optimal of the primal. So maximize the lower bound:

\[
\max_{\lambda} g(\lambda) = \max_{\lambda} \left( g_1(\lambda) + g_2(\lambda) \right)
\]

- The lagrangian dual \( g(\lambda) \) is always a concave function of \( \lambda \), so \(-g(\lambda)\) is convex, and hence is minimized easily.
- Sub-gradient descent is used to optimize wrt \( \lambda \).
∀\( \lambda \), \( g(\lambda) \) is a lower bound on the optimal of the primal. So maximize the lower bound:

\[
\max_{\lambda} g(\lambda) = \max_{\lambda} \left( g_1(\lambda) + g_2(\lambda) \right)
\]

- The lagrangian dual \( g(\lambda) \) is always a concave function of \( \lambda \), so \(-g(\lambda)\) is convex, and hence is minimized easily.
- Sub-gradient descent is used to optimize wrt \( \lambda \).
- Subgradient of \( g_1(\lambda) = \min_{y_1} \left( f_1(y_1) + y_1^T \lambda \right) \) is given by
  \[
  \nabla(g_1(\lambda)) = \bar{y}_1 = \arg \min_{y_1} \left( f_1(y_1) + y_1^T \lambda \right)
  \]
  . So computing subgradient essentially involves solving this minimization.
Maximizing Dual: Algorithm

\[
\max_\lambda \left( g_1(\lambda) + g_2(\lambda) \right) = \max_\lambda \left[ \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) + \min_{y_2} \left( f_2(y_2) - \lambda^T y_2 \right) \right]
\]

Initialize \( \lambda \) (can be arbitrary).

2. Compute subgradient of \( g_1(\lambda) \) and \( g_2(\lambda) \). Computing subgradient involves solving \( \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) \) which is doable because \( f_1(y) \) and \( f_2(y) \) are minimizable.

3. Use subgradient to update value of \( \lambda \) (usual gradient descent update rule).

\[
\lambda_{t+1} = \lambda_t + \alpha_t (\bar{y}_1 - \bar{y}_2)
\]

4. Goto Step 2, repeat until convergence of \( g(\lambda) \).

5. Now we have the optimal \( \lambda \), we need to recover the primal variables \( y_1 \) and \( y_2 \).
Maximizing Dual: Algorithm

$$\max_\lambda (g_1(\lambda) + g_2(\lambda)) = \max_\lambda \left[ \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) + \min_{y_2} \left( f_2(y_2) - \lambda^T y_2 \right) \right]$$

1. Initialize $\lambda$ (can be arbitrary).
Maximizing Dual: Algorithm

\[ \max_\lambda \left( g_1(\lambda) + g_2(\lambda) \right) = \max_\lambda \left[ \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) + \min_{y_2} \left( f_2(y_2) - \lambda^T y_2 \right) \right] \]

1. Initialize \( \lambda \) (can be arbitrary).
2. Compute subgradient of \( g_1(\lambda) \) and \( g_2(\lambda) \). Computing subgradient involves solving \( \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) \) which is doable because \( f_1(y) \) and \( f_2(y) \) are minimizable.
Maximizing Dual: Algorithm

$$\max_{\lambda} \left( g_1(\lambda) + g_2(\lambda) \right) = \max_{\lambda} \left[ \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) + \min_{y_2} \left( f_2(y_2) - \lambda^T y_2 \right) \right]$$

1. Initialize $\lambda$ (can be arbitrary).
2. Compute subgradient of $g_1(\lambda)$ and $g_2(\lambda)$. Computing subgradient involves solving $\min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right)$ which is doable because $f_1(y)$ and $f_2(y)$ are minimizable.
3. Use subgradient to update value of $\lambda$ (usual gradient descent update rule). $\lambda_{t+1} = \lambda_t + \alpha_t (\bar{y}_1 - \bar{y}_2)$. 
Maximizing Dual: Algorithm

\[
\max_{\lambda} \left( g_1(\lambda) + g_2(\lambda) \right) = \max_{\lambda} \left[ \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) + \min_{y_2} \left( f_2(y_2) - \lambda^T y_2 \right) \right]
\]

1. Initialize \( \lambda \) (can be arbitrary).
2. Compute subgradient of \( g_1(\lambda) \) and \( g_2(\lambda) \). Computing subgradient involves solving \( \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) \) which is doable because \( f_1(y) \) and \( f_2(y) \) are minimizable.
3. Use subgradient to update value of \( \lambda \) (usual gradient descent update rule). \( \lambda_{t+1} = \lambda_t + \alpha_t (\bar{y}_1 - \bar{y}_2) \).
4. Go to Step 2, repeat until convergence of \( g(\lambda) \).
Maximizing Dual: Algorithm

\[
\max_{\lambda} \left( g_1(\lambda) + g_2(\lambda) \right) = \max_{\lambda} \left[ \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) + \min_{y_2} \left( f_2(y_2) - \lambda^T y_2 \right) \right]
\]

1. Initialize \( \lambda \) (can be arbitrary).
2. Compute subgradient of \( g_1(\lambda) \) and \( g_2(\lambda) \). Computing subgradient involves solving \( \min_{y_1} \left( f_1(y_1) + \lambda^T y_1 \right) \) which is doable because \( f_1(y) \) and \( f_2(y) \) are minimizable.
3. Use subgradient to update value of \( \lambda \) (usual gradient descent update rule). \( \lambda_{t+1} = \lambda_t + \alpha_t (\bar{y}_1 - \bar{y}_2) \).
4. Goto Step 2, repeat until convergence of \( g(\lambda) \).
5. Now we have the optimal \( \lambda \), we need to recover the primal variables \( y_1 \) and \( y_2 \).
By solving the dual we get the globally optimal dual variable $\lambda^* = \arg\max_\lambda g(\lambda)$. But we need the primal variables $(y_1, y_2)$ as that's what we are interested in optimizing originally.
By solving the dual we get the globally optimal dual variable \( \lambda^* = \arg \max_{\lambda} g(\lambda) \). But we need the primal variables \((y_1, y_2)\) as that’s what we are interested in optimizing originally.

Note that while optimizing the dual, we computed subgradients which involved minimization over primal variables. We use those solutions as our primal variables, i.e:
By solving the dual we get the globally optimal dual variable $\lambda^* = \arg \max_\lambda g(\lambda)$. But we need the primal variables $(y_1, y_2)$ as that’s what we are interested in optimizing originally.

Note that while optimizing the dual, we computed subgradients which involved minimization over primal variables. We use those solutions as our primal variables, i.e:

$$y_1^* = \arg \min_{y_1} \left( f_1(y_1) + \lambda^* T y_1 \right)$$

$$y_2^* = \arg \min_{y_2} \left( f_2(y_2) - \lambda^* T y_2 \right)$$
The obtained primal solutions $y^*_1$ and $y^*_2$ will in general not satisfy $y^*_1 = y^*_2$ (its only satisfied when duality gap is 0).
The obtained primal solutions $y_1^*$ and $y_2^*$ will in general not satisfy $y_1^* = y_2^*$ (its only satisfied when duality gap is 0).

One way is choose one of $y_1^*$ or $y_2^*$ (whichever one has lower primal value).
The obtained primal solutions $y_1^*$ and $y_2^*$ will in general not satisfy $y_1^* = y_2^*$ (its only satisfied when duality gap is 0).

One way is choose one of $y_1^*$ or $y_2^*$ (whichever one has lower primal value).

Currently I haven't figured out how people find a good feasible primal solution (Victor help!).

\[
f(y) = \theta_{ac}(y_a, y_c) + \theta_{ad}(y_a, y_d) + \theta_{ab}(y_a, y_b) + \theta_{bc}(y_b, y_c) + \theta_{bd}(y_b, y_d)
\]

\[y \in \{1, 2, \cdots, L\}\]
Examples

\[ f_1(y^1) = \theta_{ac}(y^1_a, y^1_c) + \theta_{ad}(y^1_a, y^1_d) + \frac{1}{2} \theta_{ab}(y^1_a, y^1_b) \]

\[ f_2(y^2) = \frac{1}{2} \theta_{ab}(y^2_a, y^2_b) + \theta_{bc}(y^2_b, y^2_c) + \theta_{bd}(y^2_b, y^2_d) \]

\[ f(y) = f_1(y) + f_2(y) \]
Other applications

- Applications to optimization of higher order potentials (i.e. energies involving more than just pairwise terms).
- Parallelization of optimization (next ..)
The missing parts

- It's easy to extend the discussion to decomposition into $> 2$ parts.
- There can be many choices for decomposing, which is a good choice?
- Theoretical properties, what guarantees can we get on the primal solution.