Subgradient Search Algorithm

1. Arbitrarily choose \( u^1 \geq 0 \), set \( k = 1 \) and let an initial incumbent for \( (D_L) \) be \( v^0 = -\infty \).

2. Solve the Lagrangian relaxation \( (P_{uk}) \), obtaining an optimal solution \( x^k \). If \( (b-Ax^k) \leq 0 \) and \( u^k(b-Ax^k) = 0 \), stop; \( x^k \) solves \( (P) \) and \( v(P_u) = v(D_L) = (P) \).

3. If \( v(P_{uk}) > v^{k-1} \) then set \( v^k = v(P_{uk}) \) to obtain a new incumbent for \( v(D_L) \). Otherwise set \( v^k = v^{k-1} \).

4. Find a new vector of multipliers, \( u^{k+1} \) by setting \( u^{k+1} = u^k + \lambda_k(b-Ax^k)/\|b-Ax^k\| \) where \( \lambda_k \) is a scalar that satisfies

\[
\begin{align*}
\lambda_k & \geq 0 \quad \text{for all} \ k \\
\lim_{k \to \infty} \lambda_k &= 0 \\
\sum_{k=1}^{\infty} \lambda_k &= \infty
\end{align*}
\]

5. Project \( u^{k+1} \) on \( \{u \geq 0\} \) by setting \( u^{k+1}_i = \max(0, u^{k+1}_i) \) for all \( i \). Replace \( k \) by \( k+1 \) and go to step 2.

EXAMPLE 18.9:

Consider Example 18.1 and the corresponding Figure 18.3. Our choice of \( \lambda_k \) will be \( \lambda_k = 4/2^k \) where \( k = \langle k/3 \rangle \), i.e. \( \{\lambda_k\} = \{2, 2, 2, 1, 1, 1/2, 1/2, 1/2, 1/4, 1/4, 1/4, 1/4, 1/6, \ldots\} \). For this example, decreasing \( \lambda_k \) any faster would lead to repetitions solutions and directions of change in multipliers.
Figure 18.3
Subgradient Search Algorithm for Example 18.1
e.g. $x^1 = x^2 = x^3, x^4 = x^5$, etc.

Choosing $u^1 = (6,1)$ we solve $(P_u1)$ and obtain $x^1 = x^B = (0, 0, 1, 1, 0)^t$. $\nu(P_u1) = c x^1 + u^1(b-Ax^1) = 13 + (6,1)(-4,5)^t$ which gives us an incumbent solution value for the Lagrangian dual, $\nu^1 = -6$. Since $(b-Ax^1) = (-4,5)^t$ we do not have a primal feasible and complementary solution. To obtain our next set of multipliers we set $u^2 = u^1 + \lambda_1(b-Ax^1)/||b-Ax^1||$ or $u^2 = (6,1) + 2(-4,5)/||-4,5||$ or $(4.7506, 2.5617)$. The following table and Figure 18.3 shows the results of the first fourteen iterations of the algorithm.

<table>
<thead>
<tr>
<th>Iteration (k)</th>
<th>$u^k$</th>
<th>$x^k$</th>
<th>$(b-Ax^k)^t$</th>
<th>$\nu(P_u1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(6.0000,1.0000)</td>
<td>B</td>
<td>(-4,5)</td>
<td>-6.000</td>
</tr>
<tr>
<td>2</td>
<td>(4.7506,2.5617)</td>
<td>G</td>
<td>(-3,3)</td>
<td>3.4334</td>
</tr>
<tr>
<td>3</td>
<td>(3.3364,3.9760)</td>
<td>A</td>
<td>(0,-3)</td>
<td>5.0721</td>
</tr>
<tr>
<td>4</td>
<td>(3.3364,1.9760)</td>
<td>G</td>
<td>(-3,3)</td>
<td>8.9187</td>
</tr>
<tr>
<td>5</td>
<td>(2.6293,2.6831)</td>
<td>A</td>
<td>(0,-3)</td>
<td>8.9508</td>
</tr>
<tr>
<td>6</td>
<td>(2.6293,1.6831)</td>
<td>G</td>
<td>(-3,3)</td>
<td>7.1613</td>
</tr>
<tr>
<td>7</td>
<td>(1.9222,2.3902)</td>
<td>A</td>
<td>(0,-3)</td>
<td>9.8295</td>
</tr>
<tr>
<td>8</td>
<td>(1.9222,1.8902)</td>
<td>G</td>
<td>(-3,3)</td>
<td>9.9039</td>
</tr>
<tr>
<td>9</td>
<td>(1.5686,2.2437)</td>
<td>A</td>
<td>(0,-3)</td>
<td>10.2687</td>
</tr>
<tr>
<td>10</td>
<td>(1.5686,1.7437)</td>
<td>G</td>
<td>(-3,3)</td>
<td>10.5253</td>
</tr>
<tr>
<td>11</td>
<td>(1.3919,1.9205)</td>
<td>A</td>
<td>(0,-3)</td>
<td>11.2385</td>
</tr>
<tr>
<td>12</td>
<td>(1.3919,1.6705)</td>
<td>G</td>
<td>(-3,3)</td>
<td>10.8359</td>
</tr>
<tr>
<td>13</td>
<td>(1.2151,1.8473)</td>
<td>A</td>
<td>(0,-3)</td>
<td>11.4582</td>
</tr>
<tr>
<td>14</td>
<td>(1.2151,1.6801)</td>
<td>G</td>
<td>(-3,3)</td>
<td>11.3966</td>
</tr>
</tbody>
</table>

Note that although each direction is an ascent direction since the subgradient is unique, the step size chosen leads to a
decrease in \( (P_u k) \) at \( k = 6, 12 \) and 14. Perhaps decreasing \( k \) even less after that would lead in this example to a better convergence to a small neighborhood of the optimal multipliers.

Convergence of Subgradient Search

Clearly if we stop in step 2 with an \( x \) feasible to \( (P) \) satisfying complementarity, then by Theorem 18.3, 
\[
(v_{P_u k}) = v(D_L) = v(P).
\]
If the algorithm does not stop in step 2, we will show that it generates a sequence \( \{v^k\} \) satisfying 
\[
\lim_{k \to \infty} v^k = v(D_L).
\]
Our proof closely follows that given in Parker and Rardin [ ] which is based on Poljak [1967].

Each \( v^k = \max(v^{k-1}, v(P_u k)) \) thus providing a monotone nondecreasing sequence \( \{v^k\} \) such that \( v^k \leq v(D_L) \). We will show that for any \( \hat{v} < v(D_L) \) there exists a finite \( k \) satisfying \( v^k \geq \hat{v} \). Thus \( \{v^k\} \) must converge in the limit to \( v(D_L) \).

Choose \( \hat{v} < v(D_L) \) and \( \hat{u} \geq 0 \) such that \( v(P_u) > \hat{v} \). From continuity of \( v(P_u) \) at \( \hat{u} \), there exists a \( \hat{\delta} > 0 \) such that \( v(P_u) \geq \hat{v} \) for all \( u \geq 0 \) satisfying \( ||u - \hat{u}|| < \hat{\delta} \). That is, there is a neighborhood of \( \hat{u} \) in which \( v(P_u) \geq \hat{v} \) since \( v(P_u) > \hat{v} \).

Consider the sequence of Lagrange multipliers generated by the algorithm in step 4. \( u^{k+1} \) is the projection of \( u^k + \lambda_k s^k \) on the set \( \{u \geq 0\} \). As noted earlier, this implies 
\[
||\hat{u} - u^{k+1}||^2 \leq ||\hat{u} - (u^k + \lambda_k s^k)||^2
\]
where \( s^k = (b - Ax^k)/||b - Ax^k|| \). We now expand the right side of the last inequality, then add and subtract an equal term involving \( \lambda_k \):
\[ \| \hat{u} - u^{k+1} \|^2 \leq \| u - u^k \|^2 - 2 \lambda_k s^k (u - u^k) + \lambda_k^2 \]

\[ = \| u - u^k \|^2 + \| \lambda_k s^k \|^2 - 2 \lambda_k s^k (\hat{u} - u) - \lambda_k^2 \| s^k \|^2 - u + \lambda_k^2 \]

\[ = \| u - u^k \|^2 + \| \lambda_k s^k \|^2 - 2 \lambda_k s^k (\hat{w} - u^k) - 2 \hat{\lambda} \| \lambda_k s^k \| \] \hspace{0.5cm} (18.3)

where \( \hat{w} = u - \hat{\rho} \lambda_k s^k / \| \lambda_k s^k \| \).

Note that \( \| \hat{w} - u \| = \| u - \hat{\rho} \lambda_k s^k / \| \lambda_k s^k \| = \hat{\rho} \| \lambda_k s^k \| / \| \lambda_k s^k \| = \hat{\rho} \)

so by our definition of \( \hat{\rho} \) \( v(P_{\hat{w}}) \geq \alpha \).

Now assume \( v(P_{\hat{u}}) < \alpha \) for all \( k \). Noting that \( v(P_{\hat{w}}) \geq \alpha \) we have \( v(P_u) < v(P_{\hat{w}}) \) for all \( k \). Since \( s^k \) is a subgradient of \( v(P_u) \) at \( u = u^k \), we have \( v(P_{\hat{w}}) \leq v(P_u) + s^k (\hat{w} - u^k) \). \( v(P_u) \) \( < v(P_{\hat{w}}) \) for all \( k \) implies \( s^k (\hat{w} - u^k) > 0 \). \( \lambda_k > 0 \) implies that \(-2 \lambda_k s^k (\hat{w} - u^k) < 0 \) in expression (18.3). Thus \(-2 \lambda_k s^k (\hat{w} - u^k) \) can be removed from expression (18.3). Together with noting \( \| s^k \| = 1 \) (see definition of \( s^k \)) we obtain

\[ \| \hat{u} - u^{k+1} \|^2 \leq \| u - u^k \|^2 + 2 - 2 \hat{\lambda} \lambda_k \]

\[ = \| u - u^k \|^2 - \hat{\rho} \lambda_k + \lambda_k (\lambda_k - \hat{\rho}) \] \hspace{0.5cm} (18.4)

Because \( \lambda_k \to 0 \) as \( k \to \infty \), there exists some \( \ell \) for which \( \lambda_k \leq \hat{\rho} \) for all \( k \geq \ell \) implying \( \lambda_k (\lambda_k - \hat{\rho}) \leq 0 \) for all \( k \geq \ell \). Thus we can drop the corresponding term in (18.4) and obtain

\[ \| \hat{u} - u^{k+1} \|^2 \leq \| u - u^k \|^2 - \hat{\rho} \lambda_k \quad \text{for all} \quad k \geq \ell \] \hspace{0.5cm} (18.5)

Summing both sides of (18.5) for \( k = \ell, \ell + 1, \ldots, \ell + q \) and
eliminating like terms gives us

$$||u - u_k + q + 1||_2 \leq ||u - u_k||_2 - \sum_{k=0}^{+\infty} \lambda_k$$  \hspace{1cm} (18.6)

The left side of (18.6) is nonnegative since it is a squared norm. However, the right side of (18.6) will become arbitrarily negative as \( q \to \infty \) because \( \sum \lambda_k = \infty \) by design in step 4 of the algorithm. Thus (18.6) contradicts our assumption that \( \nu(P_uk) < \hat{\alpha} \) for all \( k \). Therefore there must be some finite \( k^* \) satisfying \( \nu(P_{uk^*}) \geq \hat{\alpha} \). Recalling \( \kappa = \max (\kappa^*, \nu(P_{uk})) \) it follows that \( \nu \geq \hat{\alpha} \) and our proof is complete.

Algorithmic Options

An alternative to the stepsize rule given in step 4 (Poljak (1969), Held et al. (1973)) is given by

$$\lambda_k = \frac{\nu(z - \nu(P_k))}{||b - Ax_k||^2}$$

where \( \hat{z} \) is an underestimate of \( \nu(D_L) \), i.e. \( \hat{z} < \nu(D_L) \), and for each \( k, \epsilon < \lambda_k < 2 \) for some fixed \( \epsilon > 0 \). The sequence \( \nu(P_{uk}) \) will either converge to \( \hat{z} \) or we will obtain multipliers \( u_k \) such that \( \nu(P_{uk}) \geq \hat{z} \). This choice of step-size rule can be shown to be akin to employing the linear relaxation procedure to find a solution to a system of linear inequalities (Agmon (1954), Motzkin and Schoenberg (1954)) to the following system:

$$u(b - Ax^t) + cx^t \geq \hat{z} \hspace{1cm} t = 1, \ldots, T.$$  \hspace{1cm} (18.7)
If we replace $\hat{z}$ by the unknown $(D_L)$, Poljak [1969] shows that geometric convergence results.

An interesting note can be made by looking at subgradient optimization as simply linear relaxation applied to the system (18.7). Agmon (1954) and Motzkin and Schoenberg (1954) show that any violated inequality can be chosen to determine the next change of variables. Thus, if $\nu(P_{uk}) < \hat{z}$ we continue working in (18.7) having chosen $t$ corresponding to $k$ by solving $(P_{uk})$. That is, we choose which inequality to base the next change of multipliers on by solving $(P_{uk})$. We are either finished or else we have found an $x \in S$ which gives a violated inequality (using $u = u_k$). Now it can be seen that we do not always have to solve $(P_{uk})$ to optimality!

It is sufficient to find any $x \in S$ violating (18.7) for the current choice of $u = u_k$. Thus while solving $(P_{uk})$, if we find a feasible solution $x$ to $(P_{uk})$ that satisfies $u^k(b-Ax) + cx < \hat{z}$, we can use that $x$ to determine $u^{k+1}$. Alternatively, since an overestimate of $\nu(D_L)$ is often used for $\hat{z}$ (see discussion below) we could terminate solving $(P_{uk})$ if we find an $x \in S$ such that $u^k(b-Ax) + cx \leq \nu^k$ where $\nu^k = \max(\nu(P_{uj}))$ is the incumbent value for $\nu(D_L)$. Unless $(P_{uk})$ is trivially solved, we may save significant effort by using the above rules to "stop early" in its solution. Farwan and Rardin (1983) have used such a rule in the case of surrogate duality (see Section 18.1) and obtained significant empirical advantage.

In practice (Fisher (1981)), the most commonly used rule is to use an overestimate of $\nu(D_L)$ for a choice of $\hat{z}$. This can
be obtained by finding an incumbent solution to the primal problem, say \( x \), since \((D_L) \leq (P) \leq cx\). To approximate the conditions necessary for convergence, one then chooses \( x_k \) to satisfy the same conditions put on \( x_k \) in step 4 of the algorithm. That is, \( x_k \geq 0 \) for all \( k \), \( \lim_{k \to \infty} x_k = 0 \) and \( \sum_{k=1}^{n} x_k = \hat{z} \). Of course, in a branch and bound procedure in which one is trying to use the dual for fathoming purposes, we could choose \( \hat{z} = \nu^*(P) \), the value of the primal incumbent. Then, if we reach \( \hat{z} \) the corresponding candidate problem can be fathomed. If convergence to \( \hat{z} \) is slow, or distorted because \( \hat{z} \) may be an overestimate of \( \nu(D_L) \), then we can continue branching in \( (P) \).

Ascent Methods

We noted earlier that subgradients are not always ascent directions. Recall Example (18.8) in which the subgradients \((-3,0)^t\) and \((-3,3)^t\) were descent directions at \( u = (3, 2 2/3) \). The theorem below shows us that ascent directions are the set of directions which have positive inner products, i.e. are within 90° of all primary subgradients. Of course, if a direction has a positive inner product with a set of vectors, it has a positive inner product with all convex combinations of that set of vectors. Recalling that the set of subgradients is composed of all convex combinations of the set of primary subgradients, we can conclude that directions of ascent are precisely those directions having a positive inner product with all subgradients. Figure 18.4 shows the ascent directions for the case of Example (18.8) mentioned above.
Figure 18.4
Graph of Ascent Directions for Example 18.1 at point $u = (3, 2 \ 2/3)$. 
THEOREM 18.8

Let \((P_u), \cap(P_u), \) and \(\nabla(P_u)\) be defined as above. Then \(d\) is an ascent direction for \(\nabla(P_u)\) at a point \(\hat{u}\) if and only if 
\[d(b-Ax) > 0 \quad \text{for all } x \in \cap(P_u).\]

Proof: Since \(S\) is finite, there must exist a \(\delta > 0\) for any direction \(d\) and any \(\hat{u}\) such that
\[\Omega(P_{\hat{u}+\lambda d}) - \cap(P_u) \quad \text{for all } \lambda \in (0, \delta).\]
That is, a sufficiently small step in the direction \(d\) does not produce solutions which were not already optimal solutions to \(\nabla(P_u)\). Let \(x \in \Omega(P_{\hat{u}+\lambda d}) - (P_u)\). Thus
\[
\nabla(P_{\hat{u}+\lambda d}) - \nabla(P_u) = [c\hat{x} + (u+\lambda d)(b-A\hat{x})] - [c\hat{x} + u(b-Ax)] = d(b-Ax).
\]
(18.8).

Clearly, if \(d(b-Ax) > 0\) for all \(x \in \Omega(P_u)\) then \(\nabla(P_{\hat{u}+\lambda d}) > \nabla(P_u)\), i.e. \(d\) is an ascent direction.

Conversely, suppose \(d\) is an ascent direction and assume \(d(b-Ax) \leq 0\) for some \(x \in \cap(P_u)\). Then by (18.8) we have
\[\nabla(P_{\hat{u}+\lambda d}) \leq \nabla(P_u) \quad \text{for all } \lambda \in (0, \delta).\]
This contradicts the assumption that \(d\) is an ascent direction. Thus \(d\) must satisfy
\[d(b-Ax) > 0 \quad \text{for } x \in \cap(P_u).\]

In order to find a direction of ascent for \(\nabla(P_u)\) at \(u = \hat{u}\) we could solve the following mathematical programming problem.

Maximize \(\epsilon\)

subject to: \(d(b-Ax) \geq \epsilon \quad \text{for all } x \in \cap(P_u)\)
\[d_i \geq 0 \quad \text{for all } i \text{ with } \hat{u}_i = 0\]
\[|d| \leq 1\]

If the optimal solution has \(\epsilon = 0\), then \(\hat{u}\) must solve the Lagrangian dual since no ascent direction exists. The constraint
\[ d_i \geq 0 \text{ for all } i \text{ with } u_i = 0 \text{ guarantees an ascent direction for which we can satisfy the constraint set } \{ u \geq 0 \}. \]

If we were to ignore this condition, and compute \( d \) satisfying only \( d(b-Ax) > 0 \) we may not be able to take a positive step in the direction \( d \). If we go ahead and step in the direction \( d \) and then project back to \( \{ u \geq 0 \} \), we are not guaranteed to have improved upon \( v(P_u^+) \). Therefore our problem is set up to find an ascent direction in the projected multiplier space.

The condition \( |d| \leq 1 \) is necessary to bound \( \epsilon \) if \( z > 0 \) is feasible. Any normalization of \( d \) is sufficient and because we are maximizing the minimum \( d(b-Ax) \) over \( |d| \leq 1 \), we can say that we have found the steepest ascent direction with respect to the chosen norm. For example, \( |d| \leq 1 \) could represent the linear constraints

\[-1 \leq d_i \leq 1 \quad i=1, \ldots, m\]

or \[ -1 \leq \sum_{i=1}^{m} d_i \leq 1 \]

thus giving us a linear programming problem to find \( d \). The euclidean norm, \( ||d|| \), could be used to give us what is usually thought of as the direction of steepest ascent. However, the associated mathematical programming problem is nonlinear and hard to solve.

Given an ascent direction, \( d \), we need to decide how far to step in that direction. By construction, \( v(P_{u+d}^+) \) increases as \( u \) increases from zero, so long as the set of solutions \( 0(P_{u+d}^+) \) remains a subset of \( 0(P_u^+) \). That is,
\[ cx + (\hat{u} + c\hat{d})(b - Ax) \geq cx + \hat{u}(b - Ax) \quad \forall x \in \mathcal{P}(P_u) \]

since \( \hat{d}(b - Ax) > 0 \) for all \( x \in \mathcal{P}(P_u) \).

If we step too far in the direction \( \hat{d} \), we may enter another region that has optimal solutions not contained in \( \mathcal{P}(P_u) \). Accordingly, we cannot guarantee that \( \hat{d}(b - Ax) > 0 \) for the new \( x \not\in \mathcal{P}(P_u) \) and may thus get a decrease in \( \nu(P_u) \). In the algorithm stated below, we can think of using a bisection or Fibonacci type search on \( \alpha \) to guarantee \( \mathcal{P}(P_{u+\alpha} \hat{d}) \supseteq \mathcal{P}(P_u) \).

Choosing \( \alpha \) as large as possible that satisfies this condition will yield the largest increase in \( \nu(P_u) \) in the direction of \( \hat{d} \).

A Steepest Ascent Algorithm

The following is a steepest ascent algorithm based on the above discussion.

1. Arbitrarily choose \( u^1 \geq 0 \) and set \( k = 1 \). Also choose a suitable vector norm, e.g., a Euclidean norm.

2. Solve \( (P_{uk}) \) and obtain the set of all optimal solutions, \( \mathcal{P}(P_{uk}) \). If any \( x \in \mathcal{P}(P_{uk}) \) satisfies \( (b - Ax) \leq 0 \) and \( u^k(b - Ax) = 0 \), stop; that \( x \) solves \( (P) \) and \( \nu(P_u) = \nu(D_L) = \nu(P) \).

3. Solve the following problem

\[
\begin{align*}
\text{Maximize} & \quad c \\
\text{subject to:} & \quad \hat{d}(b - Ax) \geq c \quad \forall x \in \mathcal{P}(P_{uk}) \\
& \quad d_i \geq 0 \quad \forall i \quad \text{with} \quad u_i = c \\
& \quad |d| \leq 1
\end{align*}
\]
with optimal solution $(u^k, d^k)$.

If $e^k = 0$, stop; $u^k \in \mathcal{D}_L$ and $(D_L) = v(P_uk)$.

4. Determine $\alpha^k = \sup \{ \alpha > 0 : \Omega(P_uk + \alpha d^k) \subseteq \Omega(P_uk) \text{ and } u^k + \alpha d^k \geq 0\}$.

If $\alpha^k = \infty$, stop; $(D_L) = \infty$ implying the primal problem $(P)$ has no feasible solution. Otherwise set $u^{k+1} = u^k + \alpha^k d^k$.

Replace $k$ by $k+1$ and go to step 2.

EXAMPLE 18.10

Consider Example 18.1 and let the norm of $d$, $|d|$, in step 3 be the supremum norm given by $-1 \leq d_i \leq 1$ for all $i=1,...,m$. As in the previous example we begin with $u^1 = (6,1)$, solve $(P_{u1})$ and obtain $\Omega(P_{u1}) = x^B$ with $(b-Ax^B) = (-4,5)^t$. Since $x^B$ is not feasible to the primal problem we go to step 3 to find an ascent direction. In this example, since $B$ is the unique solution to $(P_{u1})$ we do not have to solve the linear program and choose $d^1 = (d_1, d_2) = (-4,5)$. That is, $(-4,5)$ is the gradient of $(P_u)$ at $u = u^1$ and hence is the steepest ascent direction using a Euclidean norm. We now determine $\alpha^1 = \sup \{ \alpha > 0 : \Omega(P_{u1} + \alpha d^1) \subseteq \Omega(P_{u1}) \text{ and } u^1 + \alpha d^1 \geq 0\} = 1/14$. That is, as we travel in the direction $(-4,5)$ we reach the boundary between region $B$ and $G$ with steps size $\alpha^1 = 1/14$ (see Figure 18.5). Thus we obtain $u^2 = u^1 + \alpha^1 d^1 = (6,1) + 1/14(-4,5) = (80/14, 19/14)$.

Returning to step 2 we find $\Omega(P_{u2}) = \{x^B, x^G\}$. Since neither solution is primal feasible and complementary we go to
Figure 18.5
Steepest Ascent Algorithm for Example 18.1
step 3 to find an ascent direction:

Maximize \( \varepsilon \)

subject to: 
\[-4d_1 + 5d_2 \geq \varepsilon \]
\[-3d_1 + 3d_2 \geq \varepsilon \]
\[-1 \leq d_1 \leq 1 \]
\[-1 \leq d_2 \leq 1 \]

The solution to the above problem is \((d_1^2, d_2^2) = (-1,1)\) with \(\varepsilon = 6\). To determine our step size we find

\[ \sup \{ \alpha > 0 : \Omega(Pu^2 + \alpha d^2) \subseteq \Omega(Pu^2) \text{ and } u^2 + \alpha d^2 \geq 0 \} = 16/9. \]

That is, as we travel in the direction \((-1,1)\), we reach the boundary between region \(G\) and \(A\) with step size \(\alpha^2 = 16/9\) (see Figure 18.5). Thus we obtain

\[ u^3 = u^2 + \alpha^2 d^2 = (80/14, 19/14) + 16/9(-1,1) \]
\[ = (248/63, 395/126) = (3.59/63, 3.17/126) \]

Returning to step 2 we find \(\Omega(Pu^3) = \{x^A, x^G\}\) with \((b-Ax^A) = (0,-3)^t\) and \((b-Ax^G) = (-3,3)^t\). Neither solution solves the primal so we find an ascent direction in solving the following linear program:

Maximize \( \varepsilon \)

subject to: 
\[-3d_1 + 3d_2 \geq \varepsilon \]
\[-1 \leq d_1 \leq 1 \]
\[-1 \leq d_2 \leq 1 \]
with solution \((d^3_1, d^3_2) = (-1, -0.5)\) and \(\epsilon^3 = 3/2\).

The stepsize \(\epsilon\) in the direction \(d^3\) is determined by
\[
\sup \{ \epsilon > 0 : \epsilon \left( P_{u^3} + \epsilon d^3 \right) \subset \left( P_{u^3} \right) \text{ and } u^3 + \epsilon d^3 \geq 0 \} = 220/63.
\]
As we travel in the direction \((-1, -0.5)\) from \(u^3\) we reach the G, A, C boundary with a stepsize of \(220/73\). Our new multipliers are \(u^4 = u^3 + \frac{3}{2} d^3 = (4/9, 25/18)\).

After solving \((P_{u^4})\) we obtain \(\Omega(P_{u^4}) = \{x^A, x^G, x^C\}\) and need to solve one more linear program to show that \(u^4 = u^* \subset \Omega(D_L)\). Noting that \((b-Ax^C) = (1, 1)^t\) we solve:

Maximize \(\epsilon\)
subject to:
\[
\begin{align*}
d_1 - 3d_2 & \geq \epsilon \\
-3d_1 + 3d_2 & \geq \epsilon \\
1d_1 + 1d_2 & \geq \epsilon \\
-1 & \leq d_1 \leq 1 \\
-1 & \leq d_2 \leq 1
\end{align*}
\]
with solution \(d^4 = (0, 0)\) and \(\epsilon^4 = 0\). Thus no ascent direction exists and \(u^4\) solves \((D_L)\) with \(v(P_{u^4}) = v(D_L)\).

Convergence of The Steepest Ascent Algorithm

As in the case of the earlier algorithms, if we stop in step 2, we have \(v(P_{u^k}) = v(D_L) = v(P)\) via Theorem 18.3.

In step 3, if \(\alpha^k > 0\), we strictly improve upon \(v(P_{u^k})\). However \(\alpha^k = 0\) implies that no ascent direction exists from \(u^k\) so \(v(P_{u^k}) = v(D_L)\).

Termination in step 4 with \((P)\) infeasible is justified by noting that \(v(P_{u^k, \epsilon^k}) \leq v(D_L) \leq v(P)\) for all \(u^k, \epsilon, \epsilon^k\) and