Supplementary Handout on Integer Programming **Vincent Crawford**
Economics 172A **Winter 2008**

General Observations

Integer restrictions occur naturally in many settings:

Location decisions

Mutually exclusive alternatives

> ($\Sigma y_i = 1$, $y_i = 0$ or 1, $i = 1,\ldots,n$)

Contingent decisions

> ($y_j \leq y_k$, $y_i = 0$ or 1, $i = j,k$)

K of your choice out of N constraints must hold

> ($a_1 x \leq b_1 + My_1$, $a_2 x \leq b_2 + My_2,\ldots,$ $\Sigma y_i = N\text{-}K$)

Functions with your choice among N values

> ($d(x) = \Sigma d_i(x)y_i$, $\Sigma y_i = 1$, $y_i = 0$ or 1, $i = 1,\ldots,n$)

Set-up costs/fixed charges

> (replace $f_i(x_i) = c_i + d_i x_i$ if $x_i > 0$, 0 if $x_i = 0$ by $c_i y_i + d_i x_i$, $x_i \leq My_i$, $y_i = 0$ or 1)

Other nonconvexities

> (approximate by several tangent lines with set-up costs, with constraints that allow the program to "use" at most one of them; such approximation is also possible for separable convex problems, but for them no set-up costs are needed)

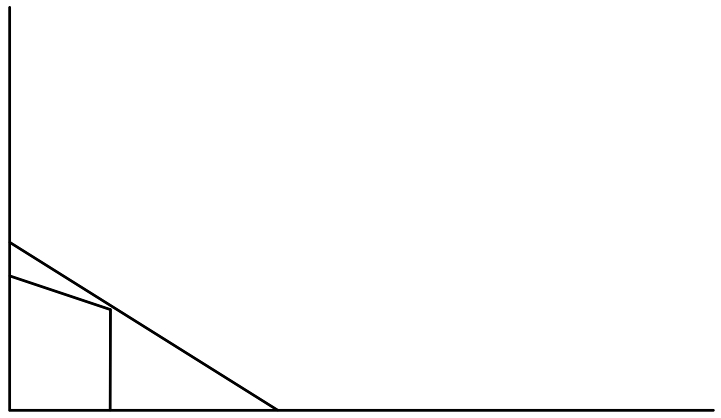
In general, solutions to problems ignoring integer restrictions need not be close to solutions respecting them, though there are important exceptions (assignment and transportation problems, problems with integer restrictions on only one variable)

Choose x_1 and x_2 to solve maximize $x_1 + 6x_2$ s.t. $x_1 \leq 3$

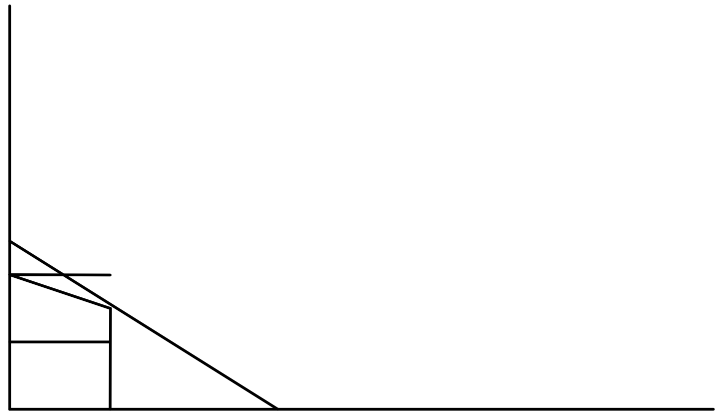
$x_1 + 10x_2 \leq 20$

$x_1 \geq 0$, $x_2 \geq 0$

Ignoring integer restrictions, solution $x_1^* = 3$, $x_2^* = 1.7$ (x_2 on vertical axis).



But when both x_1 and x_2 are required to be integers, the solution "jumps" to $x_1 = 0$, $x_2 = 2$. You can see this by shifting the objective function contour downward in the graph:



Optimal Assignment Problem and the Hungarian Method (see also Section 3 of Prof. Sobel's Notes VIII. The Transportation Problem)

Primal assignment problem (all sums are from 1 to n):

Choose x_{ij} (i person, j job) 0 or 1 to solve

$$\min \Sigma\Sigma c_{ij}x_{ij}$$

subject to $\Sigma x_{ij} \geq 1$, j = 1,…,n (all jobs filled)
$\Sigma x_{ij} \leq 1$, i = 1,…,n (no person overused)

Dual (all sums are from 1 to n, and all constraints hold for all i,j):

Choose u_i, v_j unrestricted to solve

$$\max \Sigma u_i + \Sigma v_j$$

subject to $u_i + v_j \leq c_{ij}$

(Primal constraints must hold with equality for all feasible assignments, so dual shadow prices u_i, v_j are unrestricted.)

Can solve assignment problem as a standard linear program or transportation problem using the simplex method or the transportation simplex method, but the special structure allows a more powerful and illuminating method.

Hungarian method (named for mathematicians König and Egervary)

Example to show mechanics: four people, four jobs, costs all ≥ 0

	1	2	3	4
A	94	1	54	68
B	74	10	88	82
C	62	88	8	76
D	11	74	81	21

Row reduction (subtract smallest number in each row from all numbers in that row, obtaining a new reduced cost matrix):

	1	2	3	4
A	93	0	53	67
B	64	0	78	72
C	54	80	0	68
D	0	63	70	10

Column reduction (subtract smallest number in each column from all numbers in that column, obtaining a new reduced cost matrix):

	1	2	3	4
A	93	0	53	57
B	64	0	78	62
C	54	80	0	58
D	0	63	70	0

	1	2^+	3	4
A	93	0	53	57
B	64	0*	78	62
C^+	54	80	0*	58
D^+	0*	63	70	0

Next, find a maximal set of k independent zeros and a minimal cover of k lines, for example the zeros with *s and rows and columns with $^+$s.

(There might be more than one way to do this. Any way will work.)

The fact that these three zeros can all be covered by only three lines shows that there cannot be more than three independent zeros in the matrix.

(The Handout on the Hungarian Method for the assignment problem on the web page has on its second page an algorithm that always finds a maximal set of independent zeros and a minimal cover with the same number of lines, so you can always do this. But it’s easy to do without the algorithm even in large problems and you are not required to know that part.)

	1	2^+	3	4
A	93	0	53	57
B	64	0*	78	62
C^+	54	80^{++}	0*	58
D^+	0*	63^{++}	70	0

Next, subtract the smallest uncovered cost entry from all uncovered entries, and add that same number to all double-covered entries ($^{++}$). The smallest uncovered entry is 53. The new matrix is:

	1	2^+	3^+	4
A	40	0	0	4
B	11	0*	25	9
C	54	133	0*	58
D^+	0*	116^{++}	70^{++}	0

Again find a maximal set of k independent zeros and a minimal cover of k lines, for example the zeros with *s and rows and columns with $^+$s.

Again subtract the smallest uncovered cost entry from all uncovered entries, and add that number to all double-covered entries. The smallest uncovered entry is now 4. The new matrix is:

	1	2	3	4
A	36	0	0	0*
B	7	0*	25	5
C	50	133	0*	54
D	0*	120	74	0

We now have four independent zeros, which identify an optimal assignment.

The total cost can be calculated from the original matrix: 11+10 + 8 + 68 = 97.

	1	2	3	4
A	94	1	54	68*
B	74	10*	88	82
C	62	88	8*	76
D	11*	74	81	21

Note that the optimal assignment doesn't always give a job to the person who is best at it: person B gets job 2 even though person A is better at it.

Neither does it always give a person the job he is best at: person A is best at job 2 but gets job 4, at which he is awful (though not as awful as B or C).

One might have expected person D to get job 4, at which he is best; but it's optimal to give D job 1, at which everyone else is really awful: The optimal assignment is determined by comparative advantage, not absolute advantage.

(The idea of comparative advantage here is close to the idea as used in the theory of international trade, but while both refer to properties of solutions to linear programming problems (or the competitive markets that implicitly solve them in Ricardian trade theory), the models are different and the two notions of comparative advantage are not mathematically equivalent.)

Using duality theory to explain why the Hungarian method works

Primal assignment problem (all sums are from 1 to n):

Choose x_{ij} (i denotes a person, j a job) 0 or 1 to solve $\min \Sigma\Sigma c_{ij}x_{ij}$

subject to $\Sigma x_{ij} \geq 1$, $j = 1,\ldots,n$ (all jobs filled)
$\Sigma x_{ij} \leq 1$, $i = 1,\ldots,n$ (no person overused)

Dual (all sums are from 1 to n, and all constraints hold for all i, j):

Choose u_i, v_j unrestricted to solve $\max \Sigma u_i + \Sigma v_j$

subject to $u_i + v_j \leq c_{ij}$

(Primal constraints must hold with equality for all feasible assignments, so dual shadow prices u_i, v_j are unrestricted.)

Duality Theorem for assignment problem:

$\Sigma u_i + \Sigma v_j \leq \Sigma\Sigma c_{ij}x_{ij}$ for all primal-feasible x_{ij} and dual-feasible u_i and v_j.

Proof: $u_i + v_j \leq c_{ij}$, so $\Sigma\Sigma(u_i+v_j)x_{ij} = \Sigma u_i\Sigma x_{ij} + \Sigma v_j\Sigma x_{ij} = \Sigma u_i + \Sigma v_j \leq \Sigma\Sigma c_{ij}x_{ij}$.

Corollary: If primal-feasible x_{ij} and dual-feasible u_i and v_j have equal objective function values, both are optimal.

Complementary Slackness Theorem: If all n persons can be assigned to jobs such that $x_{ij} = 1$ implies $u_i + v_j = c_{ij}$ for some dual-feasible u_i and v_j, then the assignment is optimal.

Proof: Since $x_{ij} = 0$ if not 1, $\Sigma\Sigma c_{ij}x_{ij} = \Sigma\Sigma(u_i+v_j)x_{ij} = \Sigma u_i\Sigma x_{ij} + \Sigma v_j\Sigma x_{ij} = \Sigma u_i + \Sigma v_j$. Then use the Duality Theorem and the Corollary.

Initial $u_i = \min_k c_{ik}$ (ith row minimum)

Initial $v_j = \min_h [c_{hj} - u_h]$ (jth column minimum of {column - u_i})

Lemma: Initial u_i and v_j are dual-feasible. (Note: Ignore nonnegativity here.)

Proof: $u_i + v_j = u_i + \min_h [c_{hj} - u_h] \leq u_i + (c_{ij} - u_i) = c_{ij}$.

After row-reduction and column reduction, entries in reduced cost matrix are $c_{ij} - u_i - v_j \geq 0$, dual-feasible; and zeros (*) are entries where there's no slack:

	1	2^+	3	4
A	93	0	53	57
B	64	0*	78	62
C^+	54	80^{++}	0*	58
D^+	0*	63^{++}	70	0

The minimal cover of a single vertical line in column 2 and two horizontal lines in columns C and D covers all entries with no slack in the dual constraint. The uncovered entries all have slack of at least 53.

Raising the u_i of uncovered rows by 53 and lowering the v_j of covered columns by 53 removes slack but preserves dual-feasibility: It increases slack in double-covered entries by 53, it doesn't change slack in single-covered entries (either neither u_i nor v_j changes, or both change in opposite directions), and it doesn't take too much slack from uncovered entries.

More generally, raising the u_i of uncovered rows and lowering the v_j of covered columns by the smallest uncovered entry S raises the dual objective function $\Sigma u_i + \Sigma v_j$ by S[(n-r)-c)], where r and c are the numbers of row and column lines. When—after a finite number of steps—r + c = n, we are done.

Section 3 of Prof. Sobel's Notes X. Integer Programming, linked in Section C of the reading list under "Integer Programming", discusses using the branch and bound method to solve linear programs with some integer restrictions on the variables.

Here is Professor Sobel's general statement of the branch and bound method for this case (with some comments added):

1. Set v_ = $-\infty$. (v_ is the current lower bound on how well you know how to do so far in a maximization problem)

2. Bound the original problem by solving the "relaxed" problem (ignoring the integer constraints) and rounding the value down to the nearest integer.

3. If the solution to the relaxed problem satisfies the integer constraints, stop. You have a solution to the original problem. Otherwise, call the original problem a "remaining subproblem" and go to Step 4.

4. Among all remaining subproblems, select the one created most recently. If more than one has been created most recently, pick the one with the larger bound. If they have the same bound, pick randomly. (These rules are fine points, and are not essential. It's simplest just to branch first on the x_i with lower i.) "Branch" from this subproblem to create two new subproblems by fixing the value of the next available variable to either 0 or 1. (Professor Sobel's statement and his linear programming examples all have the x_i restricted to be either 0 or 1. You can use the method for more general integer restrictions, branching by adding constraints like (i) $x_i \geq k$ in one branch and (ii) $x_i \leq k$ in the other. See below.)

5. For each new subproblem, obtain its bound z by solving a relaxed version of the subproblem and rounding the value down to the nearest integer (if the relaxed solution is not an integer).

6. Attempt to "fathom" each new subproblem. A subproblem is fathomed if one of three things happens:

(a) Its relaxation is not feasible (so it cannot possibly help).

(b) Its relaxed value is less than or equal to v_ (so it cannot help).

(c) Its relaxation has an integer solution (so it could help, but you don't need to know any more about it to know if it does).

All subproblems that are not fathomed are "remaining subproblems".

7. If a subproblem is fathomed because its relaxation has an integer solution, update v_ by setting it equal to the largest of the old value of v_ and the value of the relaxed subproblem. Call a subproblem that attains v_ the "incumbent" or "candidate" solution.

8. If there are no remaining subproblems, stop. The incumbent or candidate solution is optimal. (If you stop and there are no candidate solutions, then the original problem is not feasible.) If v_ is equal to the highest upper bound z of all remaining subproblems, stop. The candidate solution is optimal. Otherwise, return to Step 4.

Professor Sobel's Notes X. Integer Programming, linked in Section C under "Integer Programming", give an example using the branch and bound method to solve the "knapsack problem"; see his pp. 2-4 and 8.

Example: Integer linear programming by the branch and bound method

You are asked to do the integer linear programming used above to show that rounding doesn't work by the branch and bound method in PS 2 # 2(d). Here I will do a slightly different example (with a completely different solution).

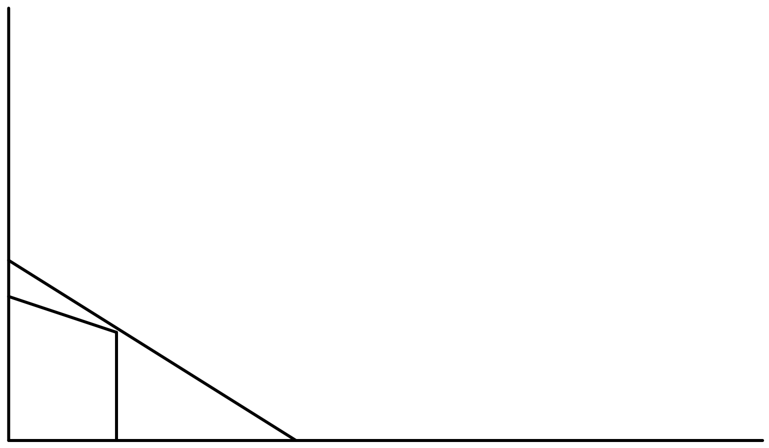
Consider the problem:

Choose integers x_1 and x_2 to solve maximize $x_1 + 5x_2$ s.t. $2x_1 \le 3$

$$x_1 + 10x_2 \le 20$$

$$x_1 \ge 0,\ x_2 \ge 0$$

v_ = −∞. Start by solving the "relaxed" problem with no integer restrictions, graphically (with x_1 on the horizontal axis and x_2 on the vertical axis):



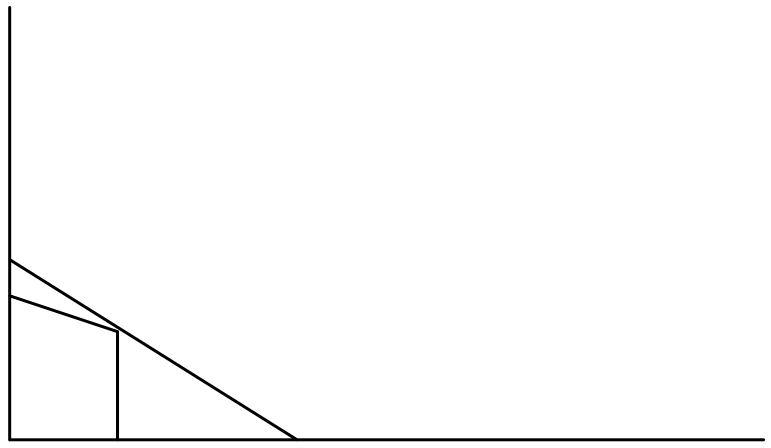
$x_1^* = 1.5$, $x_2^* = 1.85$ (from $2x_1 = 3$ and $x_1 + 10x_2 = 20$).

This solution is infeasible because it violates the integer restrictions, so the original problem is unfathomed, a "remaining subproblem." v_ = −∞ still, and there is no incumbent solution yet.

Branch on the first variable (as the x_i are numbered) that is restricted to be an integer, and isn't an integer in the solution to the relaxed problem: $x_1 = 1.5$. Branch by creating two subproblems from the original problem, one with added constraint (i) $x_1 \le 1$ and one with added constraint (ii) $x_1 \ge 2$. You could also just pick randomly, or using bounds, but do it this way here.

Choose integers x_1 and x_2 to solve maximize $x_1 + 5x_2$ s.t.

$$\begin{aligned} 2x_1 &\leq 3 \\ x_1 + 10x_2 &\leq 20 \\ x_1 \geq 0,\ x_2 &\geq 0 \end{aligned}$$



Next, attempt to fathom the two branch subproblems, starting with the first, by solving their relaxed versions.

Solving branch subproblem (i) $x_1 \leq 1$ yields $x_1 = 1$, $x_2 = 1.9$; this is infeasible because it violates the integer restrictions. $v\_ = -\infty$ still.

Branch subproblem (ii) $x_1 \geq 2$ is infeasible (because $2x_1 \leq 3$), so this branch is fathomed. $v\_ = -\infty$ still.

Further branching in subproblem (i) $x_1 \leq 1$ yields (ia) $x_1 \leq 1$, $x_2 \leq 1$ and (ib) $x_1 \leq 1$, $x_2 \geq 2$.

Solving branch subproblem (ia) yields $x_1 = 1$, $x_2 = 1$; this is feasible because it satisfies the integer restrictions, so this branch is fathomed; because there is no incumbent solution yet, $x_1 = 1$, $x_2 = 1$ becomes the incumbent solution, with objective function value 6, a lower bound on attainable values $v\_ = 6$.

[Corrected: Solving branch subproblem (ib) yields $x_1 = 0$, $x_2 = 2$; this is feasible, so this branch is fathomed; its objective function value is $10 > 6$, so $x_1 = 0$, $x_2 = 2$ is the new incumbent solution and $v\_ = 10$.

Because all branches have now been fathomed, the latest incumbent solution $x_1 = 0$, $x_2 = 2$ is optimal.]

Another example: Optimal assignment problem by the branch and bound method (this is computationally very inefficient, given the excellent alternatives available; I use it only to illustrate the method and its adaptability; see also Problem 3(c) on Problem Set #2)

Same assignment problem as before:

	1	2	3	4
A	94	1	54	68
B	74	10	88	82
C	62	88	8	76
D	11	74	81	21

Set $\bar{v} = \infty$. ($\bar{v}$ is the current upper bound on how "well" you know how to do so far in a minimization problem; note the adaptation of the algorithm.)

Solve the relaxed version of the problem in which you can fill each job with whomever you wish, without regard to duplication. (You could also do this by assigning each worker however you wish. Note the adaptation of the algorithm, with a new definition of "relaxation" for assignment problems.)

This yields the assignment D1, A2, C3, D4, with (hypothetical) cost 11 + 1 + 8 + 21 = 41. It's not feasible because person B is not assigned and person D has two jobs, so it yields no bound on attainable cost and nothing is yet fathomed. The entire problem is therefore a "remaining subproblem."

	1	2	3	4
A	94	1	54	68
B	74	10	88	82
C	62	88	8	76
D	11	74	81	21

Next, branch this remaining subproblem on how to fill job 1. (Again, this is not the only way to do it, but it's a natural way.) There are four new branch subproblems, using the obvious notation: A1, B1, C1, and D1.

In branch A1, solving the relaxed subproblem in which you imagine you can fill each job **other than job 1** with whomever you wish **other than person A—crossing out job 1 and person A but otherwise *not* enforcing feasibility in the rest of the problem**—yields A1, B2, C3, D4, which is feasible and yields cost 94 + 10 + 8 + 21 = 133. So branch A1 is fathomed; A1, B2, C3, D4 becomes the incumbent solution; and we reset $\bar{v}$ = 133.

In branch B1, solving the relaxed subproblem yields B1, A2, C3, D4, which is feasible and yields cost 74 + 1 + 8 + 21 = 104. So branch B1 is fathomed; B1, A2, C3, D4 becomes the new incumbent solution; and we reset $\bar{v}$ = 104.

In branch C1, solving the relaxed subproblem yields C1, A2, A3, D4, which is infeasible and yields (hypothetical) cost 62 + 1 + 54 + 21 = 138. But because 138 > 104 (> for minimization), branch C1 is fathomed anyway.

[corrected: In branch D1, solving the relaxed subproblem yields D1, A2, C3, A4, which is infeasible and yields (hypothetical) cost bound 11 + 1 + 8 + 68 = 88 < 104. So branch D1 is a remaining subproblem.]

	1	2	3	4
A	94	1	54	68
B	74	10	88	82
C	62	88	8	76
D	11	74	81	21

D1 is the only remaining subproblem. Further branch it on how to fill job 2. There are three new branch subproblems (ruling out infeasible reassignments of D, which is only one way to do it): D1, A2; D1, B2; and D1, C2.

[corrected: In branch D1, A2, solving the relaxed subproblem **in which you imagine you can fill each job other than 1 and 2 with whomever you wish other than D and A—crossing out jobs 1 and 2 and persons D and A but otherwise *not* enforcing feasibility**—yields D1, A2, C3, C4, which is infeasible and yields (hypothetical) cost bound 11 + 1 + 8 + 21 = 41 < 104. So branch D1, A2 is a remaining subproblem.]

In branch D1, B2, solving the relaxed subproblem yields D1, B2, C3, A4, which is feasible and yields cost 11 + 10 + 8 + 68 = 97 < 104. So branch D1, B2 is fathomed; D1, B2, C3, A4 becomes the incumbent solution; and we reset $\bar{v} = 97$. Because 97 < 102, branch D1, A2 is now also fathomed.

[corrected: In branch D1, C2, solving the relaxed subproblem yields D1, C2, A3, A4, which is feasible and yields cost 11 + 88 + 54 + 68 = 221 > 97. So branch D1, C2 is fathomed. Because all remaining subproblems are now fathomed, the last incumbent solution, D1, B2, C3, A4 with total cost 97, is optimal. (This is the same answer we got with the Hungarian Method.)]