1. OPTIMIZATION AND SURROGATE CONSTRAINTS

Problem:

$$\textit{Maximize } x_O = c\,x$$

$$\textit{Subject to } \quad A\,x \leq b$$

$$\bigcup \quad \geq x \geq O$$

Alternative Notation:

$$\textit{Maximize } x_O = \sum_{j \in N} c_j\, x_j$$

$$\textit{Subject to}$$

$$\sum_{j \in N} a_{ij}\, x_j \leq b_i\,, i \in M$$

$$\bigcup_j \geq x_j \geq O\,,\ j \in N$$

$$N = \{1,\ldots,n\},\ M = \{1,\ldots,m\}$$

Example:

Maximize

$$x_0 = 8x_1 - 3x_2 + 12x_3$$

Subject to

$$\begin{aligned}
2x_1 + 5x_2 + 11x_3 &\le 160 \\
-4x_1 - 9x_2 &\le -90 \\
7x_1 + 2x_2 - x_3 &\le 100 \\
\hline
x_1 &\le 20 \\
x_2 &\le 12 \\
x_3 &\le 12 \\
\hline
x_1, x_2, x_3 &\ge 0
\end{aligned}$$

The constraints of the problem have been separated into 3 groups, showing upper and lower bounds separately. These bounds are not included in forming surrogate constraints because they are included directly within the surrogate constraint problems, as part of the problem information.

Weight vectors for Examples on Page 4:

1. $\omega = (1 \quad 1 \quad 2)$ 2. $\omega = (1 \quad 1 \quad 1)$
3. $\omega = (1 \quad 2 \quad 1)$ 4. $\omega = (2 \quad 1 \quad 1)$

Surrogate Constraints

Select a Weight Vector

$$w = w_1, \ldots, w_m \geq 0$$

The constraint is:

$$(wA)x \leq wb)$$

or

$$\sum_{j \in N} \left(\sum_{i \in M} w_i a_{ij} \right) x_j \leq \sum_{i \in M} w_i b_i$$

or

$$\sum_{j \in N} a_{oj} x_j \leq b_o$$

$$For \quad a_{oj} = \sum_{i \in M} w_i a_{ij}$$

$$b_o = \sum_{i \in M} w_i b_i$$

Examples

A. Identify the surrogate constraint for each w vector on page 2.

B. Identify the Surrogate Constraint Problem for each of these surrogate constraints; the problem where the constraints $Ax \leq b$ are replaced by $(wA)x \leq wb$.

C. Guess an optimal solution to each problem.

(Variables here are allowed to be continuous – i.e. they do not have to be integer – i.e. they can take "fractional" rather than "whole number" values.)

Conversion for Surrogate Constraints – To Solve Easier

Use upper bounds to make all c_j of c nonnegative.

Example: Consider the surrogate problem

for w = (1 1 1). From page 2 we have:

$$\textit{Maximize } x_o = 8x_1 - 3x_2 + 12x_3$$

$$5x_1 - 2x_2 + 10x_3 \le 170$$

To change $c_2 = -3$ *to* $\bar{c}_2 \ge 0$:

$$x_2 \le 12 \rightarrow x_2 + y_2 = 12,\ y_2 \ge 0$$

$$y_2 = 12 - x_2 \ge 0$$

$$x_2 = 12 - y_2 \ge 0$$

(using x_2 bound information from page 2)

Substitute for x_2 from expression above:

$$\textit{Max } x_o = 8x_1 - 3(12 - y_2) + 12x_3$$

$$5x_1 - 2(12 - y_2) + 10x_3 \le 170$$

What does this give after clearing terms?

Exercise 1.1:

A. Guess an optimal solution to this new problem (where y_2 replaces x_2).
B. What value results for x_2?
C. How does this compare to previous guess? (From Examples, Part C, on page 4.)

Exercise 1.2:

Convert each of the other surrogate constraint problems.

Carry out A, B & C of Exercise 1.

Exercise 1.3:

Can you identify a rule that will always solve a single constraint LP problem like this? (Assume all variables are nonnegative and have upper bounds.)

Terminology: A single constraint LP is called a *Continuous Knapsack Problem*.

When the variables are integer (i.e., required to be integer-valued), the problem is called an *Integer Knapsack Problem* – or simply a

Knapsack Problem

Exercise 1.4: For the Continuous Knapsack Problems previously examined, identify one that is stronger than the others, in the sense of giving an optimum value for x_o that is closer to the optimum value when all constraints of the original problem are considered (i.e. where the original problem itself is solved).

Hint 1: A surrogate constraint is always valid for the original problem. Why?

Hint 2: If the surrogate constraint replaces the original constraints, the problem is relaxed – the set of feasible solutions can only increase in size, or stay the same. Why?

Hint 3: The observation of Hint 2 implies that the optimum objective (x_o) value for the surrogate problem will always be "Better" ($\geq$) than for the original problem. Why?

2. STRONGEST SURROGATE CONSTRAINTS & THEIR USES

- Integer programming (IP) problems are much harder to solve than linear programming (LP) problems.

- It is valuable to have bounds on x_o for solving IP problems. It is also valuable to have bounds on other variables.

- Surrogate constraints can be used to give such bounds.

- Stronger surrogate constraints give <u>tighter</u> (more restrictive) bounds, and hence are more useful for bounding.

<u>NOTE</u>: How would you define a stronger surrogate constraint for an IP problem? (How is the definition similar to, and also different from, the definition for an LP problem?)

Additional Surrogate Constraint Features For IP Problems

- They can be used to suggest "Trial Values" for integer variables.
- They can be used as source constraints for cutting planes, i.e. for additional constraints implied by the integer restrictions.
- Cutting plane inequalities, or "cuts," can be added to the original constraints to generate new (and possibly better) surrogate constraints.

Additional Uses Will be Considered Soon

Exercise 2.1: Apply the ideas of the exercises of Section 1 to reformulate the example problem in the form

$$\begin{aligned} &\textit{Minimize } \bar{x}_o = \bar{c}x \\ &\textit{subject to} \quad \bar{A}x \geq \bar{b} \\ &\qquad\qquad \bigcup \geq x \geq 0 \\ &\textit{where } \bar{x}_o = -x_o,\ \bar{c} = -c \\ &\qquad\qquad \bar{A} = -A,\ \bar{b} = -b. \end{aligned}$$

Then transform the problem so that $\bar{c} \geq o$, introducing "y variables" as appropriate.

Finally, identify the surrogate constraints for the four w vectors of shown on page 2.

A. Demonstrate these are equivalent to the original surrogate constraints.

B. Identify optimal solutions to the resulting surrogate problems.

Exercise 2.2:

How does the rule for solving a continuous knapsack problem of the form

$$\textit{Minimize} \quad \bar{c}\,x$$

$$\textit{Subject to} \quad \bar{a}_o x \geq \bar{b}_o\,, \quad \bigcup \geq x \geq 0,$$

$$\textit{where } \bar{a}_o = w\bar{A}, \quad \bar{b}_o = w\bar{b},$$

compare to the rule for solving the earlier "maximize" form?

Heuristic Methods for Knapsack Problems

It is useful to have a very fast method to obtain "good" solutions to Knapsack Problems. Such a heuristic method, which has the goal of obtaining

an optimal or near-optimal solution in many instances (but is not guaranteed to do so), is often based on the same ratio calculations used to solve LP Knapsack Problems.

Exercise 2.3: Formulate an explicit set of rules, expressed as a set of instructions or a flow diagram or a pseudo code, for two different heuristics for a Knapsack Problem in "Maximizing Form." Do the same for a Knapsack Problem in "Minimize Form."

Illustrative Heuristics (for the "Maximize Form")

Heuristic 1 (Classical "Greedy" Knapsack Heuristic)

1. Order the variables so that

$$c_1 / a_{o1} \geq c_2 / a_{o2} \geq \ldots \geq c_n / a_{on}$$

(These ratios are sometimes called "Bang-for-Buck" ratios.)

Define $J = \{j \in N:\ x_j = 1\}$, where to begin J = ϕ. Finally, let $b_o' = b_o$.

2. Let $jnext = Min(j \in N - J: a_{oj} \leq b_o')$

3. If *jnext* does not exist (i.e., J = N or $a_{oj} > b_o'$ for all $j \in N - J$) then stop. Otherwise, add *jnext* to *J*, set $b_o' := b_o' - a_{jnext}$, and return to step 2.

(Note, each *jnext* added to *J* is larger than the one added before.)

Exercise 2.4 Apply Heuristic 1 to the Knapsack Problem where

$c = (13 \quad 10 \quad 17 \quad 20 \quad 8 \quad 10 \quad 4 \quad 2)$

$a_o = (\ 4 \quad 5 \quad 9 \quad 11 \quad 6 \quad 8 \quad 4 \quad 2)$

$b_o = 19.$

Improved Heuristic 1: A simple improvement of Heuristic 1 is as follows. When the method stops, assuming $J \neq \phi$, let *jmax* = max ($j \in J$). (Hence, *jmax* was the last index added to J.) Remove *jmax* from J, setting $b_o' := b_o' + a_{oj\max}$, and let *jbest* be an index $j \geq$ *jmax* that gives the largest c_j value subject

to $a_{oj} \leq b'_o$. Then add *jbest* to J. (At "worst," *jbest* = *jmax*, which is the same solution already found.)

Exercise 2.5: Apply the Improved Heuristic 1 to the problem of Exercise 2.4.

Exercise 2.6: Transform the problem of Exercise 2.4 into a minimization problem (i.e., with the "Minimize Form"). Write instructions for a version of Heuristic 1 (including its Improved Variant) that applies to this "Minimize Form," and which will yield a solution that is the same (after converting the variables back to their original identities) as the solution obtained by Heuristic 1 for the "Maximize Form."

For our next heuristic we require some notation.

For any number (value) v, define:

$\lfloor$ v $\rfloor$ = the largest integer $\leq$ v

$\lceil$ v $\rceil$ = the smallest integer $\geq$ v

Example 1: $\lfloor 2.3 \rfloor = 2$, $\lfloor 2 \rfloor = 2$, $\lfloor -2.3 \rfloor = -3$.

Example 2: $\lceil 2.3 \rceil = 3$, $\lceil 2 \rceil = 2$, $\lceil -2.3 \rceil = -2$

Define the *effective coefficient* b_{oj} associated with a_{oj} by $b_{oj} = b_o / \lfloor b_o / a_{oj} \rfloor$. Note that $b_o = b_{oj} / \lfloor b_o / a_{oj} \rfloor$, hence $\lfloor b_o / b_{oj} \rfloor = \lfloor b_o / a_{oj} \rfloor$ and this b_{oj} identifies the "effective size" of a_{oj} in terms of the integer number of times that a_{oj} divides b_o.

Heuristic 2A. (Greedy Method Using Effective Coefficients). The method is the same as Heuristic 1, except that the variables are ordered so that $c_1 / b_{o1} \geq c_2 / b_{o2} \geq \ldots \geq c_n / b_{on}$.

Define the *effective profit* for variable x_j by $c_j \lfloor b_o / a_{oj} \rfloor$. Note, this is the profit that would result if x_j could be assigned the integer value $\lfloor b_o / a_{oj} \rfloor$, which would be possible if x_j was not bounded to satisfy $x_j \leq 1$. Hereafter, we call $\lfloor b_o / a_{oj} \rfloor$ the *effective multiple* for x_j, and denote it by m_j. Hence the effective profit for x_j is $c_j m_j$.

Heuristic 2B. (Greedy Method Using Effective Profits). The method is the same as Heuristic 1, except that the variables are ordered in descending order of effective profits.

Exercise 2.7: Show algebraically that Heuristic 2A and Heuristic 2B are identical (and that their Improved Variants are also identical).

Exercise 2.8: Apply Heuristic 2B to the Knapsack Problem of Exercise 2.4. Also apply the Improved Variant.

Exercise 2.9: Carry out the instructions of Exercise 2.6, replacing Heuristic 1 by Heuristic 2B. (However, the solution need not be the same as for the "Maximization.")

For our final illustrative heuristic, define the *updated effective multiple* m'_j to be the same as m_j, except that the updated value b'_o replaces b_o, i.e.,

$m'_j = \left\lfloor b'_o / a_{oj} \right\rfloor$. Correspondingly, define the *updated effective profit* to be $c_j m'_j$.

Heuristic 3: (Greedy Method Using Updated Effective Profits.) The method is the same as Heuristic 2B, except that the variables are ordered in descending order of updated effective profits. Since the updated effective profits change each time b'_o changes, instead of pre-ordering the variables, the choice of *jnext* is given by

$c_{jnext}\ m'_{jnext} = Max\ (c_j m'_j : j \in N - J \text{ and } a_{oj} \leq b'_o)$.

Exercise 2.10: Apply Heuristic 3 to the problem of Exercise 2.4.

Exercise 2.11: Carry out the instructions of Exercise 2.9, replacing Heuristic 2B by Heuristic 3.

Exercise 2.12: As a rule, which of the three knapsack heuristics do you think is likely to be best? Can you construct three different examples, where Heuristic 1 works best on the first, Heuristic 2B works best on the second, and Heuristic 3 works best on the third?

3. HOW TO CREATE GOOD SURROGATE CONSTRAINTS

This section helps to develop intuition about normalizing original constraints – as one way to create weights to produce surrogate constraints. Later, page 41 gives normalization rules.

I. Intuitive Normalization Methods

Consider the following three systems of constraints.

$$A.\quad \begin{aligned} 20x_1 + 30x_2 + 40x_3 &\le 250 \\ 400x_1 + 200x_2 + 300x_3 &\le 2500 \\ 3x_1 + 2x_2 + 4x_3 &\le 25 \end{aligned}$$

$$B.\quad \begin{aligned} 1x_1 + 1x_2 &\le 1 \\ 1x_1 + 1x_3 + 1x_4 &\le 1 \\ 1x_2 + 1x_3 + 1x_4 + 1x_5 &\le 1 \end{aligned}$$

$$C.\quad \begin{aligned} 1x_1 + 1x_2 &\ge 1 \\ 1x_1 + 1x_3 + 1x_4 &\ge 1 \\ 1x_2 + 1x_3 + 1x_4 + 1x_5 &\ge 1 \end{aligned}$$

Exercise 3.1: Identify possible ways to "normalize" the constraints of systems A, B & C (multiply or divide each constraint by a positive quantity) so that, if the normalized constraints are summed, each one contributes an appropriate influence to the resulting surrogate constraint.

Exercise 3.2:

(1) How might you normalize inequalities of the following system

$$\begin{aligned} 1x_1 - 1x_2 &\leq O \\ 1x_1 + 1x_3 - 1x_4 &\leq O \\ -1x_2 + 1x_3 - 1x_4 + 1x_5 &\leq -1 \end{aligned}$$

$$\text{where } 1 \geq x_j \geq O \quad \text{for} \quad j = 1,\ldots,5$$

(2) Same question, but reverse the direction of the constraint inequalities.

(3) How does your answer compare to the answer to Exercise 3.1 for system B? Is that system related to this one?

Exercise 3.3: How would you normalize…?

$$
\begin{aligned}
1x_1 + 1x_2 &\geq 1 \\
1x_1 + 1x_3 + 1x_4 &\geq 2 \\
1x_2 + 1x_3 + 1x_4 + 1x_5 &\geq 3 \\
\textit{where } 1 \geq x_j \geq O \quad \textit{for} \quad j &= 1,\ldots,5
\end{aligned}
$$

Compare your answer to that for Exercise 3.1, for system B. Should these answers be related?

4. Multidimensional 0-1 Knapsack Problem

Consider the problem

$$\textit{Maximize} \quad x_O = cx$$

$$Ax \leq b$$

$$x \textit{ is binary}$$

in detached coefficient form

	x
c	
A	b

Example

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	b
C=	36	83	59	11	43	67	23	52	93	25	
A=	5	9	17	23	10	12	11	15	12	15	57
	3	15	5	15	11	9	15	17	10	25	61
	13	21	11	25	12	23	15	12	9	10	65

A problem such as this, where c > O, b > O and

A ≥ O, is called a Multidimensional Knapsack

Problem.

<u>Exercise 4.1</u>. Create 3 different surrogate constraints:

(1) $w = (1 \quad 1 \quad 1)$,

(2) $w = (2 \quad 1 \quad 1)$,

(3) $w = (1 \quad 1 \quad 3)$

Exercise 4.2:

Use two Heuristic Rules (your favorites, subject to not being too complex or hard to apply), to generate 0-1 solutions for the surrogate constraint Knapsack Problems from Exercise 4.1.

(A) Check each solution to see if it is feasible for the original problem — or to see how much it violates feasibility.

(B) Based on (A) and on the value of x_o, pick the 2 solutions you like best.

Exercise 4.3: Transform the original problem into the form

$$\begin{aligned} \textit{Minimize } y_o &= \bar{c}\ y \\ \bar{A}\, y &\geq \bar{b} \\ y &\textit{ is binary} \end{aligned}$$

using $y_j = 1 - x_j$, $j \in N$.

Exercise 4.4: Apply Exercise 4.1 to the transformed problem, except let $w_2 = 3$ in (2) and (3).

Exercise 4.5: Apply Exercise 4.2 to the surrogate constraint Knapsack Problems of Exercise 4.4.

Exercise 4.6: Compare the best solutions from exercises 4.2 and 4.5. Are they the same after being expressed in terms of x?

MORE ADVANCED HEURISTICS

Exercise 4.7:

Start with the best infeasible solution from previous exercises.

Now consider setting $x_j = O$ for any x_j that is 1 in this best solution but = O in the 2nd best solution. Do this one variable at a time, using any choice rule you want to choose such an x_j from available possibilities. Repeat until obtaining a feasible solution. (If not enough variables can be changed from 1 to O by this rule to make the best solutions become feasible, continue by considering the 3rd best solution as a source for $x_j = O$.)

Remark: This is a special case of the evolutionary approach for combining solutions called *path relinking*.

Exercise 4.8:

Apply Exercise 4.7, but reverse the roles of the best and 2^{nd} best solutions.

Conditional Surrogate Constraints

For the following, chose a (single) rule for generating a surrogate constraint, $w_i = 1/b_i$ for each $i \in M$.

Exercise 4.9: Use 2 rules to generate a heuristic solution, but then let the rules "vote" by picking

(A) $x_j = 1$ for a variable x_j that both rules select for $x_j = 1$, if possible.

(B) Given the priority of (A), chose $x_j = 1$ for the x_j that has $x_j = 1$ in at least one (preferably both) solution, and which has the largest bang for buck ratio.

- Do (A) and (B) just to choose a single best $x_j = 1$ assignment. Then, set this $x_j = 1$ and thereby remove this x_j from the problem. (This changes b and removes a column from A.)
- For each new (smaller) problem, repeat the process.
- As soon as choosing $x_j = 1$ produces an infeasible solution, remove this and all other variables that will create an infeasible solution

by setting $x_j = 1$. (Is there an easy way to identify them?) Then do (A) and (B) just for remaining variables.

- Once it is necessary to remove one or more x_j variables as just described, continue to remove such variables that create infeasible solutions on each subsequent step.
- Finally, when no x_j can be set to 1, return to the last step where a feasible solution was produced by setting $x_j = 1$. If this x_j does not have the largest c_j among all x_j that can be set to 1, then, change this choice by picking instead $x_j = 1$ for this largest c_j. (Note: the solution is feasible.)

Exercise 4.10: Compare this solution to the best feasible solution found in exercises 4.7 & 4.8.

Exercise 4.11: List ways to reduce the computation effort of a strategy like Exercise 9. (What change in the strategy might give about the same result, but faster? When might it be ok to choose $x_j = 1$ for more than a single x_j?)

Exercise 4.12: How might you change the strategy of Exercise 4.9 if you used more than one surrogate constraint to generate solutions? (Since different surrogate constraints give different "Bang for Buck" ratios, how would you modify (B) of Exercise 4.9? Also, given that there are more than two trial solutions to "vote" on setting $x_j = 1$, how else might you change the "priorities" of voting?)

Exercise 4.13: Use the minimization formulation of Exercise 4.3, and then apply a "counterpart" of Exercise 4.9 to this formulation. (How do some of the instructions of Exercise 4.9 change? Write down the changed instructions.)

5. Adaptive Generation of Surrogate Constraints

Problem Reduction by Logical Implications

It is often valuable to check problem constraints and surrogate constraints for logical implications that can simplify and "reduce" the problem. These logical implications can be checked both before undertaking to solve the problem, and also at each step after assigning a value to a variable in a constructive (or destructive) heuristic process. (Depending on the problem and the stage of solution, several variables may be selected and assigned values between two successive steps of checking for logical implications.)

For convenience, the following describes the logical implication tests for the problem in its original form, before assigning values to any of the variables, since the updated problem after making such assignments has this same form.

We consider the 0-1 problem using both the "$\leq$ representation" and the "$\geq$ representation," where all constraint coefficients $\left(a_{ij}\right)$ are assumed non-negative. As already seen, any constraint can be given either of these representations and the non-negativity assumption can be assured by the usual transformations. Define $sum(i) = \sum_{j \in N} a_{ij}$.

Zero-One Logical Implications for "≤ Constraints":

(A) $\sum_{j \in N} a_{ij} x_j \leq b_i$

(all $a_{ij} \geq O$, all x_j binary)

---

1. If $b_i < O$, the problem has no feasible solution.
2. If $sum(i) \leq b_i$, the constraint is redundant.
3. If $a_{ij} > b_i$, then $x_j = O$.

Zero-One Logical Implications for "≥ Constraints"

$$\text{(B)} \qquad \sum_{j \in N} a_{ij} x_j \geq b_i$$

$$(\text{all } a_{ij} \geq O, \text{ all } x_j \text{ binary})$$

---

1. If $sum(i) < b_i$, the problem has no feasible solution.
2. If $b_i \leq O$, the constraint is redundant.
3. If $a_{ij} > sum(i) - b_i$, then $x_j = 1$.

Note that it is easy to keep updated values of *sum(i)* (as well as of b_i) as variables are selected and assigned values, to facilitate checking for associated logical conditions. Removing redundant

constraints from consideration can be important for determining surrogate constraints, since redundant constraints should receive a O weight. (Such constraints are not permanently removed from consideration, of course, but should be reinstated if the partial solution that created their redundancy is changed.)

Linked Implications

Surrogate constraints can be subjected to the preceding tests as readily as other problem constraints. If a variable is compelled to receive a particular value by one of the preceding tests, then it is possible that this may uncover new implications ─ by reference to a constraint that has not been checked since the variable was assigned

such a value. Consequently, it can be useful to re-check a constraint for a possible logical implication if a variable was assigned a value since the last time the constraint was checked.

Note: Stronger logical implications than the ones illustrated above, making use of bounded sums of variables, are given in the original 1965 surrogate constraint paper, and implications of additional strength using nested sums are given in the 1971 paper "Flows in Arborescences." (References appear at the end of these notes.)

We now review the rules for creating surrogate constraints by normalizations, i.e., where assigning a normalized constraint a weight of 1 automatically implies an associated weight to be assigned to the

form of the constraint that is not subjected to normalization. The rules differ according to whether the constraint has the form of (A) or (B) in the description of the tests for logical implications.

After examining outcomes produced by these normalizations, we consider adaptive rules for generating surrogate constraints.

Type 1 Normalization Rules:

The weight w_i for normalizing is given by

$$w_i = (\sum a_{ij} - b_i) / b_i \qquad \text{for (A)}$$

$$w_i = b_i / (\sum a_{ij} - b_i) \qquad \text{for (B)}$$

These rules are symmetric, in that they yield the same normalization regardless of whether the original constraint is put in the form of (A) or (B) (by complementing variables, as necessary).

The next rules considered are not symmetric. (They are not necessarily better or worse than the Type 1 rules, in general. But in certain settings the Type 1 rules can be shown to be preferable.)

Type 2 Normalization Rules:

Normalize (A) and (B) by first dividing through by b_i. Then, representing the new coefficients as if they were the original ones, and for some power $k \geq 1$, define

$$D = \left(\sum_{j \in N} a_{ij} \right)^k$$

(i.e., the coefficients above are those after dividing by b_i). Then, to complete the normalization:

Rule 1, for (A): Multiply through by D.

Rule 2, for (B): Divide through by D.

Exercise 5.1: Identify w_i for the Type 2 Nomalization Rules 1 and 2. Show algebraically that the outcomes of Rules 1 and 2 are similar but not identical to each other. Compare the outcomes to the expressions for w_i in the Type 1 rules.

Exercise 5.2: In the Type 2 rules, choose k = 1 and 2, and in each case apply Rules 1 and 2 to normalize the ≤ form and the ≥ form of each constraint in system B at the beginning of Section 3. Show that the relative sizes of w_i for the 3 constraints are similar by Rules 1 and 2, but not precisely the same. Apply the Type 1 rules in the

same way and compare their w_i values to those of the Type 2 rules.

Basis for Adaptive Weighting

Let x* be a candidate solution for the problem, not necessarily feasible. (For example, x* may be obtained by a surrogate constraint heuristic, with or without updating.) If x* is infeasible, and the surrogate constraint

$$\sum_{j \in N} a_{oj} x_j \quad \leq b_o$$

was used to generate x*, then increase the weights on constraints violated by x*.

Exercise 5.3: What type of rule(s) might be used to do this? (weights on satisfied constraints might also be decreased. How would your rule(s) change in this case?) Demonstrate your rules by a numerical example.

Exercise 5.4: If you generate a new x* for the new surrogate constraint ─ i.e., by a heuristic or algorithm using this constraint and repeating the process ─ then:

- Can you use info from previous applications of the process to help choose (or restrict the choice of) the weights?
- How would you decide when to stop?
- Which surrogate constraint would you choose?

Again, demonstrate by a numerical example.

An Adaptive Surrogate Constraint Method

Suppose x* is "representative" of the kinds of trial solutions produced by a particular method you are using. For example, x^* might be a weighted average of the r best of these trial solutions (for r = 10, 20, 30, etc.) with a threshold t such that any x_j^* that is greater than t is rounded to give $x_j^* = 1$. For present purposes, it is appropriate to set t small, including the possibility t = 0 (in which case

$x_j^* = 1$ results if $x_j = 1$ in any solution x that is included in the average).

Then, instead of normalizing the i th constraint by dividing or multiplying through by $(\sum_{j \in N} a_{ij})^k$, divide or multiply through by

$$\left(\sum_{j \in N} a_{ij} x_j^* \right)^k .$$

This may give a more "realistic" set of normalizations. Explain why. (Note that the previous normalization results from the new "adaptive" one if x* is the vector of all 1's)

Exercise 5.5: In a method that updates the problem information at each step of assigning a value to a

variable, how might the basis for choosing x^* also change at each step? (Or periodically?)

Exercise 5.6: What benefits could result from having several different surrogate constraints? In your answer consider constraints that result both by choosing different values of the power k and by choosing different ways of determining x^*. How would you use more than one surrogate constraint to make decisions about values to assign to variables?

Additional "Frequency" Memory

Consider an approach that keeps track of feasible "good" solutions x(1), x(2),…,x(H), and for each solution x(h), keeps track of

$$s_i(h) = b_i - \sum_{j \in N} a_{ij}\, x_j(h) \text{ for each } i.$$

That is, $s_i(h)$ is the "slack value" for constraint i in solution $x(h)$.

Exercise 5.7. How might you use the $s_i(h)$ values to weight constraints? What is the reason for your rule?

Exercise 5.8. Should your approach of Exercise 5.7 also include other influences — as illustrated from previous exercises — to determine weights? Which ones?

Exercise 5.9. Illustrate how you would implement your ideas of Exercises 5.6, 5.7 and 5.8, by applying them to the multidimensional Knapsack Problem shown in Section 4.

REFERENCES:

F. Glover (1965). "A Multiphase-Dual Algorithm for Zero-One Integer Programming Problems," *Operations Research*, Vol. 13, No. 6, pp. 8789-893.

F. Glover (1971). "Flows in Arborescences," *Management Science*, Vol. 17, No. 0, pp. 568-586.