Incremental Pruning - Technical Notes

Anthony R. Cassandra

January 6, 1997

Abstract

This paper provides analyses and discussions on the Incremental Pruning IP algorithm for solving POMDPs exactly.

1 Introduction and notation

The *cross-sum* of two sets of vectors is the set of all possible combinations of a vector from the first set and a vector from the second set. We we denote this operation with the operator $\oplus$.

The sets S_z^a are the transformations of the previous step's $(t-1)$ value function planes for action a and observation z. It is the value of a belief state at time t of following the $t-1$ policy given that we perform action a and observe z. We assume discounting, if present, is incorporated into these values.

The Incremental Pruning algorithm (IP) does incremental cross sums on the S_z^a sets. There are a number of ways to do the cross sums and a number of ways to decide on the cross-sum ordering. These will all be discussed in this paper.

We have two methods for doing the individual cross-sums. The normal version just cross-sums and does normal Lark filtering; I call this Normal Cross-Sum (NCS). The other variation, Restricted Region (RR), tries to be a bit smarter about generating vectors in the cross-sum.

For both cross-sum variations we can use the *save-a-point* option. This uses a set of points to help initialize the set of vectors we are trying to produce. The NCS with the save-a-point option, is denoted by NCS-SP. For RR, we assume we always save a point. Since there is no extra cost is acquiring a point and since each point saves an LP, considering the case of RR without saving a point is pointless.

We will analyze the variations from botom to top, so that each analysis can build on the others. We will use two types of analysis throughout: total number of linear programs, LPs; and total number of LP constraints.

Some of the comparisons compare average case, some best case and some worst case. I am a little leary of a direct comparison of any of these. Comparing one worst case agains another worst case may not be that valid, especially if one of the worst cases is not even achievable. I should distinguish between achievable and unachievable best and worst cases, since this seems important.

2 LP analysis of Lark's filtering algorithm.

All the cross-sum variations require the lark filtering algorithm in some form. The Lark filtering algorithm takes two sets as input: the set of useful vectors and the set of vectors to be filtered. The set of useful vectors could be empty or could be initialized with any number of vectors. We break the analysis of this algorithm into three parameters and the complexity is defined in terms of these. The first parameter is I which is the size of the initial set. The second parameter is U, which is the number of vectors to be filtered that will not be in the final solution. The last parameter, Y, is the number of vectors to be filtered that are useful.

2.1 Lark filtering - total LPs

The number of LPs is easy to determine, it is $U+Y$, since we must set up one LP for each vector that needs to be checked.

$$\mathrm{LF}_{\mathrm{LPs}}(I,U,Y) = U + Y \tag{1}$$

This is the same for best, worst, average and any other case you could devise.

2.2 Lark filtering - total constraints

There are at least three ways to analyze this: worst case, best case and average case. I ignore the number of variables, since it is the same for a given problem (i.e., the number of states). It is tempting to ignore the simplex constraint, since it is only one constraint and is common to all the cases, but many the different algorithms may require differing number of LPs. This means that one algorithm would be ignoring more constraints than another, which means that it is necesary to include the simplex constraint in our analyses.

There is also the non-negativity constraint on all variables, but this it not usually something that needs to be explicitly stated in the LP. In fact, it is usually necessary to go out of your way if you want the variables to assume negative values.

2.2.1 Lark Filtering - total constraints - best case

Since the number of constraints in an LP is directly related to the number of vectors we started with, I, and the number of useful vectors found so far, the best case is when all of the I initial vectors dominate all of the useless vectors and we happen to selected all the useless vectors before processing the useful vectors. This keeps the size of the LPs as small as possible for as long as possible.

We start with I vectors, so our first LP will have this many constraints plus the one simplex constraint. In addition, all LPs up until we find our first useful vector will have this number of constraints. After that, all subsequent LPs will be one constraint larger than the previous as we add more and more useful LPs.

$$\begin{aligned}\mathrm{LF}_{\mathrm{best}}(I,U,Y) &= \sum_{i=1}^{U}(I+1) + \sum_{i=1}^{Y}(I+i) \\ &= IY + U(I+1) + \frac{Y(Y+1)}{2}\end{aligned} \tag{2}$$

This is the best case total number of constraints over all LPs in a single application of the Lark filtering algorithm.

We note that if $I=0$, the best case of eqaution 2 is not actually achievable. In this case, the first LP is guaranteed to find a point that leads to a useful vector. Thus

$$\begin{aligned}\mathrm{LF}_{\mathrm{best}}(0,U,Y) &= 1 + \mathrm{LF}_{\mathrm{best}}(1,U,Y-1) \\ &= 1 + \sum_{i=1}^{U} 2 + \sum_{i=1}^{Y-1}(i+1) \\ &= Y + 2U + \frac{Y(Y-1)}{2}\end{aligned} \tag{3}$$

2.2.2 Lark Filtering - total constraints - worst case

When we happen to select all the useless vectors after all the useful vectors, then when we get to the useless vectors the LPs are as large as they possibly can be, thus worst case. Note that this is more likely than the best case, since the best case requires not only that we select all the useless vectors first, but also that the initial set of N vectors completely dominate all the useless vectors.

The main difference between this and the best case, is the summation for the number of constraints in the LPs processed for the useless vectors.

$$\begin{aligned} \mathrm{LF}_{\mathrm{worst}}(I,U,Y) &= \sum_{i=1}^{Y}(I+i) + \sum_{i=1}^{U}(I+Y+1) \\ &= UY + IY + U(I+1) + \frac{Y(Y+1)}{2} \end{aligned} \tag{4}$$

The worst case is just the best case number of constraints plus an extra UY constraints.

2.2.3 Lark Filtering - total constraints - (unrealistic) average case

Discussion of the average case has to begin with exactly what is the average case. For this, I assume that at any given step, each of the remaining vectors are equally likely to be chosen; we either choose a useful or a useless vector. However, here we also make the unrealistic assumption that when we select a useless vector, it is completely dominated by all the vectors we have already found. See section 2.2.4 for discussion of the more realistic case.

Therefore, given these assumptions, the size of the subsequent LP in the sequence of LPs will depend upon whether we selected a useful or useless vector. We simply weight each by the probability of selecting each type of vector. This is easily stated with the recurrence:

$$\mathrm{LF}_{\mathrm{ave}}(i,u,y) = i + 1 + \frac{u}{u+y}C(i,u-1,y) + \frac{y}{u+y}C(i+1,u,y-1)$$

with base cases:

$$\begin{aligned} \mathrm{LF}_{\mathrm{ave}}(i,0,0) &= 0 \\ \mathrm{LF}_{\mathrm{ave}}(i,0,y) &= i+1+C(i+1,0,y-1) \\ \mathrm{LF}_{\mathrm{ave}}(i,u,0) &= i+1+C(i,u-1,0) \end{aligned}$$

Since the worst case is just the best case plus some number of constraints, we would expect the average case to have the same structure. By comparing the best case to the average case empirically, we were able to deduce that the average case number of constraints was exactly halfway between the best case and worst case. In other words the closed form for this recursion is

$$\mathrm{LF}_{\mathrm{ave}}(I,U,Y) = \frac{UY}{2} + IY + U(I+1) + \frac{Y(Y+1)}{2} \tag{5}$$

which can be verified inductively (see appendix A). This is somewhat surprising, since we mentioned that the worst case and best case were not equally likely.

2.2.4 Lark Filtering - total constraints - average case

To more accurately define the average case, we need to consider, not whether we select a useful vector or not, but whether we select a vector that leads to a useful vector. Note that a useless vector can lead to finding a useful vector, if the useless vector is not completely dominated by vectors already in the list, since it will give us a point with which we can get a useful vector.

The problem with such an analysis is that the order we select vectors and shape of the value function becomes very important and hard to define. We would need to say something about the probability that we select a vector that leads to a useful vector and this would involve saying something about distributions over value functions and vectors.

However, the average case (equation 5) is a lower bound on the true average case, since removing the assumption that a useless vector never leads to a useful vector will only serve to increase the number of constraints faster. The worst case (equation 4) is an upper upper bound on the average, thus we would expect the true average case to be somewhere between the worst case and average case. Since all the previous cases different by some factor of UY, we can parameterize the real average case with

$$\mathrm{LF}_{\mathrm{real}}(I, U, Y) = \sigma UY + IY + U(I+1) + \frac{Y(Y+1)}{2} \tag{6}$$

where $1/2 < \sigma < 1$, but otherwise unknown at this point.

3 Cross-sum Analysis

Here we discuss the case of a single cross sum operation $C = A \oplus B$ where we want to parsimonious representation of C, which we will call C^*. The cross sum operator is both associative and commutative. We will define $|A| = N$ and $|B| = M$ and without loss of generality we will assume that $N \geq M$. The number of useless vectors in C is $U = NM - |C^*|$. Note that we have some constraints: $|C| = NM$, $1 \leq C^* \leq NM$, $0 \leq U \leq NM - 1$ and all of these values are non-negative. If we assume that the sets A and B themselves are parsimonious, then we have tighter constraints so that $N \leq C^* \leq NM$ and $0 \leq U \leq NM - N$.

We note that the size of C^* cannot be known in advance, thus all the anlysis uses U in their formulas. This shows how the algorithms are affected by the number of useless vectors present.

The analysis of the Lark filtering algorithm leads to the analysis for all variations of the cross-sum algorithms (NCS, NCS-SP and RR). We will use equation 2, equation 4, equation 5 or equation 6 depending upon how we want to analyze each algorithm.

The NCS algorithm for doing cross sums takes two sets, A and B, computes the cross-sum and then uses Lark filtering to remove useless vectors. One variable in this algorithm is how to initialize the set that is sent to the Lark filtering algorithm. The plain NCS version uses the simplex corner points to select out useful vectors from the full cross sum. The enhancement NCS-SP initializes this set using points that were saved previously to find vectors in the full cross-sum.

The RR algorithm for doing cross-sums is a bit more complex, but the important part we emphasize here is that the order of the sets matter. The cross-sum operator is commutative, but when implemented with the RR algorithm, the complexity changes depending on whether the smallest or largest set is chosen to be the REGION SET. We discuss this more in the analysis of RR.

3.1 Normal Cross Sum (NCS)

When using the simplex corners to generate the initial set for Lark filtering we can assume that we get at least two vectors. If we get one vector, then that vector must dominate at all of the simplex corners and thus is must dominate everywhere: we ignore this simple case. Thus the initialize set size is 2, the number of useless vectors U, and the number of useful vectors to be discovered is $Y = NM - U - 2$.

3.1.1 NCS - total LPs

This analysis comes directly from the Lark filtering analysis

$$\begin{aligned} \mathrm{NCS}_{\mathrm{LPs}}(N, M, U) &= \mathrm{LF}_{\mathrm{LPs}}(2, U, NM - U - 2) \\ &= NM - 2 \end{aligned} \tag{7}$$

This is the worst case scenario.

3.1.2 NCS - total constraints - best case

The best case here can be defined two ways: we can get the best case with respect to the initial set size of 2; or we can measure it using the best possible initial set size.

The best case is often not that useful, since it just gives us a lower bound on the number of constraints. Thus, the best case should give us the ultimate possible best case and this is achieved by assuming the initial set size is as large as possible and that the Lark filtering algorithm goes the best way possible.

The largest the initial set size can be is when every simplex corner yields a different vector. Thus, if we let S be the number of states, then the best case definition is

$$\mathrm{NCS}_{\mathrm{best}}(N, M, U) = \mathrm{LF}_{\mathrm{best}}(S, U, NM - U - S) \tag{8}$$

or

$$\mathrm{NCS}_{\mathrm{best}}(N, M, U) = \mathrm{LF}_{\mathrm{best}}(2, U, NM - U - 2) \tag{9}$$

3.1.3 NCS - total constraints - worst case

For the worst case, the initialization in this case will only detect 2 vectors. Thus the worst case for NCS is

$$\begin{aligned}
\mathrm{NCS}_{\mathrm{worst}}(N, M, U) &= \mathrm{LF}_{\mathrm{worst}}(2, U, NM - U - 2) \\
&= U(NM - U - 2) + 2(NM - U - 2) + 3U + \frac{1}{2}(NM - U - 2)(NM - U - 1) \\
&= \frac{1}{2}\left[N^2M^2 + NM - U^2 + U - 6\right]
\end{aligned} \tag{10}$$

3.1.4 NCS - total constraints - average case

Just like the best case, we have two choices for defining the average case: average with respect to 2 initial vectors; or the average with respect to the average initial set size. We can't really define how many of the simplex corners will lead to useful vectors, but one guess (which is probably conservative for real problems) would be $S/2$. Thus we get either

$$\mathrm{NCS}_{\mathrm{ave}}(N, M, U) = \mathrm{LF}_{\mathrm{ave}}(2, U, NM - U - 2) \tag{11}$$

or

$$\mathrm{NCS}_{\mathrm{ave}}(N, M, U) = \mathrm{LF}_{\mathrm{ave}}(S/2, U, NM - U - S/2) \tag{12}$$

3.2 Normal Cross Sum with save-a-point (NCS-SP)

For this variation on the cross-sum of the sets A and B, we assume that every vector in A and B has a *witness* point associated with it. How these points are obtained is discussed in section 13. The presence of such points assumes that each vector is actual useful, or in other words, that A and B are parsimonious. The NCS analysis did not hinge on whether or not the sets were parsimonious, but here we make the assumption that they are.

With the parsimonious assumption, and the save-a-point option, we have a lower bound on the initial set size. Using the points associated with set A, since $|A| = N$ and we assume $|A| \geq |B|$, the initial set size must be at least N. We note that we could also use the points associated with B as well, but there is no way to guarantee that more than N vectors will be found. An often useful assumption is that we also use the simplex corners, since this makes the initial set size for NCS-SP, $\max(N, S)$, which is at least as big as the initial set size used in NCS.

3.2.1 NCS-SP - total LPs

Assuming that we are restricted to having exactly N vectors in the initial set, the total number of LPs that are needed is

$$\begin{aligned}
\mathrm{NCS}_{\mathrm{LPs}}(N, M, U) &= \mathrm{LF}_{\mathrm{LPs}}(N, U, NM - U - N) \\
&= NM - N
\end{aligned} \tag{13}$$

3.2.2 NCS-SP - total constraints - best case

The absolute best case could be defined for the case where each witness point and every simplex corner produced a unique useful vector. Alternatively, we could simply stay with the N vector initial size. Thus we get

$$
\begin{aligned}
\mathrm{NCSSP}_{\mathrm{best}}(N,M,U) =& \mathrm{LF}_{\mathrm{best}}(N+M+S,U,NM-U-N-M-S) \\
& \text{or} \\
\mathrm{NCSSP}_{\mathrm{best}}(N,M,U) =& \mathrm{LF}_{\mathrm{best}}(N,U,NM-U-N) \\
=& N(NM-U-N)+UN+U+\frac{1}{2}(NM-U-N)(NM-U-N+1) \\
=& N^2M+UN-N^2+UN+U \\
& +\frac{1}{2}[N^2M^2-UNM-N^2M+NM-UNM+U^2 \\
& \quad +UN-U-N^2M+UN+N^2-N] \\
=& UN-UNM+\frac{1}{2}\left[N^2M^2+NM+U^2+U-N^2-N\right]
\end{aligned}
\tag{14, 15}
$$

3.2.3 NCS-SP - total constraints - worst case

We are guranteed to get N vectors in the initial set, so the worst case is found by simply substituting into equation 4

$$
\begin{aligned}
\mathrm{NCSSP}_{\mathrm{worst}}(N,M,U) =& \mathrm{LF}_{\mathrm{worst}}(N,U,NM-U-N) \\
=& U(NM-U-N)+N(NM-U-N)+U(N+1) \\
& +\frac{1}{2}(NM-U-N)(NM-U-N+1) \\
=& \frac{1}{2}\left(N^2M^2+NM-U^2+U-N^2-N\right)
\end{aligned}
\tag{16}
$$

3.2.4 NCSP-SP - total constraints - average case

Restricting ourselves the the case where the initial set size is N, we simply substitute $Y = NM-U-N$ into equation 5 and simplify to get

$$
\begin{aligned}
\mathrm{NCSSP}_{\mathrm{ave}}(N,M,U) &= \mathrm{LF}_{\mathrm{ave}}(N,U,NM-U-N) \\
&= \frac{1}{2}\left(N^2M^2-N^2-UNM+NM+UN+U-N\right)
\end{aligned}
\tag{17}
$$

3.3 Restricted Region (RR)

Here we undertake an analysis for RR that is similar to what was done in the Lark filtering algorithm. RR requires a slightly modified version of the Lark filtering algorithm, though we will be able to apply the formulas we derived in section 2. We assume we are going to do a restricted region Lark filtering on each of F regions (in other words, there are F vectors in the region set to be cross-summed). Let there be G vectors in the other set in the cross sum. We will discuss the effects of selecting ordering, $F < G$ and $F > G$, in section 4.2.

The RR algorithm sets up the region of a vector from the *region set* and attempts to find those vectors from, among the G vectors, that dominate over this region. The cross sum of this *region vector* and the set of vectors discovered from these region LPs are guaranteed to be useful vectors in the final parsimonious representation of the full cross-sum. We assume both sets in the cross-sum are parsimonious to start with and that the region set has a witness point associated with each vector. We will use the witness point to get an initial vector from the G vectors in the other set.

Thus, we are doing a Lark filtering with $G-1$ vectors and 1 initial vector for every one of F vectors in the region set. The initial vector (it is one of the G vectors) for each region is found using the witness point of the region vector. The lark filtering analysis still applies, except that every LP (there will be $G-1$ LPs for each region) now has $F-1$ extra region constraints.

For a single region, i, the total number of constraints we have is:

$$(G-1)(F-1) + \mathrm{LF}(1, U_i, V_i - 1) \tag{18}$$

Where U_i and V_i are, respectively, the number of useless and useful vectors in the region i. Note that $\forall i, U_i + V_i = G$ and $\forall i, 0 \le U_i \le G-1$. In addition we have some other facts that will come in useful

$$\sum_{i=1}^{F} V_i = V \tag{19}$$

$$\sum_{i=1}^{F} U_i = U \tag{20}$$

For each of the F regions, we need to do the number of constraints shown in equation 18.

The total number of constraints in the RR algorithm would be:

$$\mathrm{RR}(F, G, U) = F(G-1)(F-1) + \sum_{i=1}^{F} \mathrm{LF}(1, U_i, V_i - 1) \tag{21}$$

This formula can be used to derive best, worst or average case formulas for the RR algorithm by using the appropriate substitution for $\mathrm{LF}(1, U_i, V_i - 1)$. We re-emphasize that the order of the sets matters in equation 21.

3.3.1 RR - total LPs

For each of the F regions, we must do $G-1$ LPs. Thus the total number of LPs is

$$\mathrm{RR}_{\mathrm{LPs}}(F, G) = FG - F \tag{22}$$

3.3.2 RR - total constraints - best case

This case is not that useful, but we provide the formula for completeness. It provides a lower bound on the number of constraints.

$$\mathrm{RR}_{\mathrm{best}}(F, G, U) = F(G-1)(F-1) + \sum_{i=1}^{F} \mathrm{LF}_{\mathrm{best}}(1, U_i, V_i - 1)$$

Put intermediate steps here.

$$= F^2G - F^2 - GU + \frac{1}{2}\left[FG^2 - FG + 3U + \frac{U^2}{F}\right] \tag{23}$$

3.3.3 RR - total constraints - worst case

The worst case is the most interesting, since it will allow us to make the strongest statements about the algorithm in comparison with other algorithms. We start much the same as the best case derivation:

$$\mathrm{RR}_{\mathrm{worst}}(F,G,U) = F(G-1)(F-1) + \sum_{i=1}^{F} \mathrm{LF}_{\mathrm{worst}}(1,U_i,V_i-1)$$

Put intermediate steps here

$$= F^2G - F^2 + \frac{1}{2}\left[FG^2 - FG + U - \sum_{i=1}^{F} U_i^2\right] \tag{24}$$

The appearance of the U_i term is a bit hard to cope with when comparing to other algorithms which do not have this variable. However, since that summation is being subtracted from the total, we can replace it with the minimum possible value it can achieve and still preserve the validity of this worst case equation.

It turns out that for $y = \sum_{i=1}^{n} x_i$, the term $\sum_{i=1}^{n} x_i^2$ is minimized when $x_i = y/n$ (see appendix E). Therefore, using this fact and equation 20 we get $U^2/F \leq \sum_{i=1}^{F} U_i^2$

$$\mathrm{RR}_{\mathrm{worst}}(F,G,U) = F^2G - F^2 + \frac{1}{2}\left[FG^2 - FG + U - \frac{U^2}{F}\right] \tag{25}$$

3.3.4 RR - total constraints - average case

The average case is the most interesting from the implementation point of view, since it can guide us to decide which algorithm to use in general. Here we will use equation 6 for $\mathrm{LF}(1,U_i,V_i-1)$.

$$\begin{aligned}
\mathrm{RR}_{\mathrm{ave}}(F,G,U) =& F(G-1)(F-1) + \sum_{i=1}^{F} \mathrm{LF}_{\mathrm{ave}}(1,U_i,V_i-1) \\
=& F(G-1)(F-1) + \sum_{i=1}^{F}\left[\sigma U_i(V_i-1) + V_i - 1 + 2U_i + \frac{V_i(V_i-1)}{2}\right] \\
=& F(G-1)(F-1) + \sum_{i=1}^{F}\left[\sigma U_iV_i - \sigma U_i + V_i - 1 + 2U_i + \frac{1}{2}(V_i^2 - V_i)\right] \\
=& F(G-1)(F-1) + \sigma\sum_{i=1}^{F} U_iV_i - \sigma\sum_{i=1}^{F} U_i + \sum_{i=1}^{F} V_i - \sum_{i=1}^{F} 1 \\
&+ 2\sum_{i=1}^{F} U_i + \frac{1}{2}\sum_{i=1}^{F} V_i^2 - \frac{1}{2}\sum_{i=1}^{F} V_i \\
=& F(G-1)(F-1) + \sigma\sum_{i=1}^{F} U_iV_i + (2-\sigma)\sum_{i=1}^{F} U_i \\
&+ \frac{1}{2}\sum_{i=1}^{F} V_i + \frac{1}{2}\sum_{i=1}^{F} V_i^2 - \sum_{i=1}^{F} 1
\end{aligned} \tag{26}$$

Using equations 20 and 19, substituting into equation 26 and simplifying, we get

$$\mathrm{RR}_{\mathrm{ave}}(F,G,U) = F(G-1)(F-1) + \sigma\sum_{i=1}^{F} U_iV_i + (2-\sigma)U + \frac{1}{2}V + \frac{1}{2}\sum_{i=1}^{F} V_i^2 - F \tag{27}$$

Now we derive a replacement for $\sum_{i=1}^{F} V_i^2$ by using the fact that $V_i = G - U_i$

$$
\begin{aligned}
\sum_{i=1}^{F} V_i^2 &= \sum_{i=1}^{F} V_i(G - U_i) \\
&= \sum_{i=1}^{F} GV_i - \sum_{i=1}^{F} U_i V_i \\
&= GV - \sum_{i=1}^{F} U_i V_i
\end{aligned} \tag{28}
$$

Substituting equation 28 into equation 27 yields

$$
\begin{aligned}
\mathrm{RR_{ave}}(F,G,U) &= F(G-1)(F-1) + \sigma \sum_{i=1}^{F} U_i V_i + (2-\sigma)U + \frac{1}{2}V + \frac{1}{2}GV - \frac{1}{2}\sum_{i=1}^{F} U_i V_i - F \\
&= F(G-1)(F-1) - F + (2-\sigma)U + \frac{1}{2}(G+1)V + (\sigma - \frac{1}{2})\sum_{i=1}^{F} U_i V_i
\end{aligned} \tag{29}
$$

We could bound $\sum_{i=1}^{F} U_i V_i$ by $G^2/4F$, but if we take $\sigma = 1/2$ (the unrealistic average case), the summation goes away, simplifying the analysis. Recall that $\sigma = 0.5$ was the average case of Lark filtering with some slightly unrealistic assumptions. Thus, under those same assumptions and using the fact that $V = FG - U$ the average number of constraints in the RR algorithm becomes

$$
\begin{aligned}
\mathrm{RR_{ave}}(F,G,U) &= F(G-1)(F-1) - F + \frac{3}{2}U + \frac{1}{2}(G+1)V \\
&= F^2G - F^2 + U + \frac{1}{2}\left(FG^2 - UG - FG\right)
\end{aligned} \tag{30}
$$

Note that since we would expect $1/2 < \sigma < 1$, this provides a slight under-estimate of the number of constraints, thus we would expect RR to do a little worse than this under more realistic assumptions.

3.3.5 RR-NSP

Although we emphasize RR with the save-a-point option, we quickly provide some information about RR when this option is not used. Where the RR algorithm refers to the version that save a point, we will use RR-NSP to refer to the version that does not save a point.

For the total number of constraints, we only need to alter equation 21 to have one more LP per region, thus we get

$$
\mathrm{RRNSP}(F,G,U) = FG(F-1) + \sum_{i=1}^{F} \mathrm{LF}(0, U_i, V_i) \tag{31}
$$

where we would make the appropriate substitution for $\mathrm{LF}(0, U_i, V_i)$ depending upon whether we were concerned with the best, worst or average case.

RR-NSP - total LPs When we do not save a point, we cannot initialize the set sent into the Lark filtering algorithm to anything but empty. Thus we need an extra LP to find the first point to use to identify a vector. Thus we need G LPs for each of the F regions.

$$
\begin{aligned}
\mathrm{RRNSP_{LPs}}(F,G,U) &= F * \mathrm{LF_{LPs}}(0, U, G-U) \\
&= FG
\end{aligned} \tag{32}
$$

RR-NSP - total constraints - best case

$$\mathrm{RRNSP}_{\mathrm{best}}(F,G,U) = FG(F-1) + \sum_{i=1}^{F} \mathrm{LF}_{\mathrm{best}}(0,U_i,V_i)$$

$$\text{Put intermediate steps here.}$$

$$= F^2G - GU + \frac{1}{2}\left[FG^2 - FG + 3U + \frac{U^2}{F}\right] \tag{33}$$

This is simply F^2 more constraints than in the save-a-point version of RR.

RR-NSP - total constraints - worst case

$$\mathrm{RRNSPworst}(F,G,U) = FG(F-1) + \sum_{i=1}^{F} \mathrm{LFworst}(0,U_i,V_i)$$

$$\text{Put intermediate steps here.}$$

$$= F^2G + \frac{1}{2}\left[FG^2 - FG + U - \sum_{i=1}^{F} U_i^2\right]$$

$$= F^2G + \frac{1}{2}\left[FG^2 - FG + U - \frac{U^2}{F}\right] \tag{34}$$

Just as in the best case, this is simply F^2 more constraints than in the save-a-point version of RR.

RR-NSP - total constraints - average case

$$\mathrm{RRNSPave}(F,G,U) = FG(F-1) + \sum_{i=1}^{F} \mathrm{LFave}(0,U_i,V_i) \tag{35}$$

4 Cross-sum comparisons

All of these POMDP algorithms proceed by cross-summing the S_z^a sets in some order. We'll analyze the effects of this in section 5. Here we choose to focus on a single cross-sum operation.

4.1 NCS vs. NCS-SP

Both of these do $O(NM)$ LPs, and whether NCS-SP is better than NCS hinges on the initialization step. In the worst case NCS-SP can actually be worse than NCS. We can cross-sum two sets of size N and M, where $\max(N,M)$ is less than the number of unique useful vectors at the simplex corners in the parsimonious representation of the cross-sum set. A construction of such a problem is easily accomplished by having more states than witness points and ensuring that a different policy is useful for each state.

4.2 RR-S vs. RR-L

The analysis for the RR algorithm assumed that the order of the sets (i.e., which set was the region set) was predetermined. However, give two sets to cross-sum we have the freedom to choose either as the region set. Therefore we can consider the effects of whether or not we should choose the largest or smallest set as the region set. The former will be referred to as RR-L and the latter as RR-S. We use L and S for the sizes to be cross-summed and assume that $L \geq S$.

We begin by looking at the RR-NSP variations and then the normal RR variations.

4.2.1 RR-NSP-S vs. RR-NSP-L - total LPs

The total LP for these two are identical; both require exactly NM LPs, since either there are M LPs each for N regions, or N LPs each for M regions.

4.2.2 RR-NSP-S vs. RR-NSP-L - total constraints - best case

4.2.3 RR-NSP-S vs. RR-NSP-L - total constraints - worst case

Using equation 34 and letting V equal the useful set of vectors, we get the worst case total constraints

$$\mathrm{RRNSP}_{\mathrm{worst}}(F,G) = F^2G + \frac{V}{2}\left[2G - \frac{V}{F} - 1\right] \tag{36}$$

which asymptotically is $O(F^2G + VG)$. Since $\max(F,G) \leq V \leq FG$, if $G > F$ then the complexity would be $O(FG^2)$ and when $G < F$ the complexity is $O(F^2G)$. Therefore, the worst case complexity is when is $V = \Theta(FG)$ and is always a square of the largest set, regardless of which we choose for the region set. However, when $V = o(FG)$ the situation is a little more complex. If $V = \Theta(F)$ (implying $F \geq G$), then the complexity is $O(F^2G)$ and if $V = \Theta(G)$ (implying $G \geq F$) then the complexity is $O(G^2 + F^2G)$. We note that $V = O(F)$ or $V = O(G)$ must be true, since $\max(F,G) \leq V \leq FG$.

We want to say that this last asymptotic argument tells us to pick the smaller set as the region set, but I don't know if I really can from the asymptotic point of view. I should re-examine these arguments. Also, regardless of this, I think we can actually proved that RR-NSP-S *is always no worse than* RR-NSP-L*. This is certainly the case empirically and when I actually prove it will be in appendix F.*

4.2.4 RR-NSP-S vs. RR-NSP-L - total constraints - ave case

4.2.5 RR-S vs. RR-L - total LPs

The difference in the number of LPs here is simply the different between the two set sizes. Thus, RR-L will do $L - S$ fewer LPs.

4.2.6 RR-S vs. RR-L - total constraints - best case

4.2.7 RR-S vs. RR-L - total constraints - worst case

Empirical observation: For the $792,575$ *case where* $2 \leq N \leq 50$*,* $2 \leq M \leq N$ *and* $0 \leq Q \leq NM - N$*, only* 694 *of them have* RR-L *with less constraints than* RR-S *and even then,* NCS-SP *is not better than* RR-L *or* RR-S*. For the* $791,252$ *cases where* $3 \leq N \leq 50$*,* $3 \leq M \leq N$ *and* $0 \leq Q \leq NM - N$*, only 9 of them have* RR-L *with less constraints than* RR-S*.*

When completed the proof will be in appendix B.

4.2.8 RR-S vs. RR-L - total constraints - average case

The average case (assuming $\sigma = 1/2$) analysis of RR was given in equation 30. There, F is the size of the region set and G the size of the other set. Suppose we have two sets of sizes L and S, where $L > S$. Let RR-L be the RR algorithm that uses L as the region set size and RR-S that uses S. The average case complexity for RR-L is

$$\begin{aligned}\mathrm{RRL}_{\mathrm{ave}}(L,S,U) =& \mathrm{RR}_{\mathrm{ave}}(L,S,U)\\ =& L(S-1)(L-1) - L + \frac{3}{2}U + \frac{1}{2}(S+1)V\end{aligned} \tag{37}$$

and for RR-S

$$\begin{aligned}\mathrm{RRS}_{\mathrm{ave}}(S,L,U) =& \mathrm{RR}_{\mathrm{ave}}(S,L,U)\\ =& S(L-1)(S-1) - S + \frac{3}{2}U + \frac{1}{2}(L+1)V\end{aligned} \tag{38}$$

Substracting equation 38 from equation 37 yields

$$\begin{aligned}\mathrm{RRL}_{\mathsf{ave}}(L,S,U)-\mathrm{RRS}_{\mathsf{ave}}(S,L,U)=&(L-1)(S-1)(L-s)-L+S+\frac{1}{2}V(S-L)\\=&(L-S)\left[(L-1)(S-1)-\frac{1}{2}V-1\right]\end{aligned}\tag{39}$$

which represents the number of extra constraints that RR-L will have. When equation 39 is less than zero, RR-L will have less constraints than RR-S. Since we are given that $L > S$, $L - S$ is positive and we can simplify the problem to finding whether RR-L is better than RR-S to finding when this holds:

$$\begin{aligned}(L-1)(S-1)-\frac{1}{2}V-1&<0\\LS-L-S+1-\frac{1}{2}V-1&<0\\LS-\frac{1}{2}V&<L+S\\LS-\frac{1}{2}(LS-U)&<L+S\\\frac{1}{2}(LS+U)&<L+S\end{aligned}$$

This equation gives us a way to determine, based upon size, which order to cross-sum the sets in for the RR algorithm. We can show (appendix C) that for $S \geq 4$ or for $U \geq 4$ this inequality is never satisfiable (under the constraints $L > S$, $S > 0$ and $U > 0$). In fact, we can restrict this even more. We can show that for $S = 3$, RR-L is only better if the number of useless vectors is, 0 or 1. Thus, aside from these two cases, the only time RR-L will be better is when $S = 2$ and $U \leq 3$. In the rare occasions when RR-L is better than RR-S, we will save exactly $(L-2)(2-1/2U)$ constraints.

For the $S = 2$ case, we show in appendix D that the savings that RR-L has over —sc rr-s is proportional to $1/L$, so the savings diminish as the size of the set increases, so even for the case when $S = 2$ and $U \leq 3$, the savings we get is not substantial for large sets. Additionally, although we cannot know a priori how many useless vectors there are, most commonly there will be more than 3 making RR-S the clear favorite in the average case.

4.3 NCS vs. RR-NSP-S

4.3.1 NCS vs. RR-NSP-S - total LPs

The NCS scheme has 2 less LPs than —sc rr-nsp-s.

4.3.2 NCS vs. RR-NSP-S - best case

4.3.3 NCS vs. RR-NSP-S - worst case

Using the substitution $U = NM - V$ in equation 10 we get

$$\mathrm{NCS}_{\mathsf{worst}}(N,M,NM-V)=VNM+NM-\frac{V^2+V}{2}-3\tag{40}$$

which shows that NCS asymptotically is $O(VNM)$. Equation 36 and assuming $N \geq M$, we find the asymptotic complexity of RR-NSP-S to be $O(M^2N + VN)$. In the worst case when $V = \Theta(NM)$ we see that RR-NSP-S is asymptotically better than NCS by a factor of M. At the other extreme when $V = \Theta(N)$, we get $O(N^2M)$ and $O(M^2N + N^2)$ for NCS and RR-NSP-S respectively which shows that when $N = M$ and $V = \Theta(N)$, NCS is asymptotically better. However, for any other case RR-NSP-S is asymptotically better by a factor of M.

Using equations 10 and 34 we can derive an inequality to determine the conditions under which one variation will yield less constraints than the other (assuming $N \geq M$).

$$NM(2M-2N-2)+3N^2-\frac{N^2}{M}+6 \leq 0 \tag{41}$$

I have empirically evaluated this and found the following observations which I am within a whisker of actually proving:

- For $N \geq 5$, NCS is only better if $N = M\ U$ is as large as possible (i.e., $N(n-1)$).
- For $N = 4$, the only three times NCS is better is: if $M = 4$ and $U = 12$ or $U = 11$; or if $N = 4$, $M = 3$ and $U = 8$.
- For $N = 3$, both are better about half the time. Specifically, NCS better for the following cases:
  - $M = 2$ and $U = 1$
  - $M = 2$ and $U = 2$
  - $M = 2$ and $U = 3$
  - $M = 3$ and $U = 5$
  - $M = 3$ and $U = 6$
- For $N = 2$ and $M = 2$ NCS is better.

When the proof is complete it will be in appendix G.

It is also the case that as the number of useful vectors increases, the savings of RR-NSP-S over NCS increases dramatically.

Empircal observation: RR-NSP-S *never worse than* NCS *when* $3 \leq N \leq 50$, $3 \leq M \leq N$ *and* $1 \leq Q \leq NM - N - 2$.

4.3.4 NCS vs. RR-NSP-S - average case

4.4 NCS-SP vs. RR-S

Here we assume that both NCS-SP and RR start with two parsimonious sets to be cross-summed. The size of these sets will be N and M and we will simplify things by assuming that $N \geq M$. Thus the Lark filtering in the NCS-SP case starts with N vectors. The region set size for the RR-S algorithm will be M, since section 4.2 established that selecting the smaller size for the region set was preferable in both the average and worst cases.

If N is smaller than the number of unique vectors at the simplex corners, then it would be better to use the regular NCS algorithm, or if we modified NCS-SP to use the simplex corners as well, then the Lark filtering would have $\max(N, N_s)$ vectors, where N_s is the number of unique useful vectors found by checking at the simplex corners. For the remainder of this section we will assume that $N \geq N_s$.

4.4.1 NCS-SP vs. RR-S - total LPs

NCS-SP saves an LP for each vector in the larger set for a total of $NM - N$, whereas RR-S saves an LP for each vector in the smaller set for $NM - M$. Thus their difference is just the difference between the two set sizes. We note that RR-L does exactly the same number of LPs as NCS-SP.

4.4.2 NCS-SP vs. RR-S - best case

4.4.3 NCS-SP vs. RR-S - worst case

The worst case formula for NCS-SP (with $N \geq M$) is given in equation 16. Plugging in N for G and M for F into equation 24 we get the worst case formula for RR-S.

For the worst case, we can prove (see appendix H) that RR-S is always better than NCS-SP, except if $M = 2$, $N < 8$ and $U = N$.

4.4.4 NCS-SP vs. RR-S - average case

The average case formulas for NCS-SP and RR-S are given in equations 17 and 38.

Using these formulas and the substitutions $Y = NM - U - N$ and $V = NM - U$, we can state an inequality that specifies when it is better to use NCS-SP rather than RR-S. With a lot of substitution and algebraic manipulation we get the inequality

$$\frac{1}{2}N^2M^2 + M^2 + NM + UN < NM^2 + \frac{1}{2}\left(N^2M + N^2 + UNM + U + N\right) \tag{42}$$

When this inequality is satisfied, then the NCS-SP algorithm will have fewer total constraints than the RR algorithm, making it the preferred choice. It can be shown (see appendix I) that for $M \geq 2$, this inequality can only be satsifed when $N = 2$ and $U = 3$, meaning that NCS-SP will never have fewer total constraints than RR-S, except for this one special case.

5 Set selection in IP

All of the previous analysis was for a single cross-sum operation. However, the IP algorithm incrementally cross-sums all the S_z^a sets and the order in which they are chosen for cross-summing does not affect the the final solution. The plain IP algorithm can be viewed as

$$((\ldots((S_0^a \oplus S_1^a) \oplus S_2^a) \oplus \ldots) \oplus S_{Z-1}^a) \tag{43}$$

where $\oplus$ is the cross-sum operator between two sets and Z the number of observations. However, the cross-sum operator is associative and commutative, so we have many choices for ordering the cross-sums.

We have seen that the complexity of the cross-sum algorithms depend upon the sizes of the sets, so a good ordering could potentially help us. We have also seen that the cross-sum complexity is related to the number of useful/useless vectors in the cross-sum. Ideally, given the sizes of all the parsimonious cross-sums, we could compute the ordering that has the minimal number of total constraints in the LPs.

Note that this is slightly different than the *matrix-chain multiplication problem*, since unlike matrix multiplication, this operator is commutative. This means we are free to change the ordering of the sets, as well as the choosing the parenthesization. Just choosing the parenthesization alone is exponential is Z. Although we could use dynamic programming to find the optimal parenthesization, it isn't as simple as looking at all possible orderings and then compute the best parenthesization. First there are a lot of orderings (it is all possible permutations of the sets) and second, there are many redundancies in doing this that have to be dealt with.

Of course, the size of the results of all possible cross-sums cannot be known in advance, so the question becomes how can we reason about this without that knowledge. The first question to tackle is why the order should make any difference at all. We have to do all of the sets eventually anyway, so why should the order matter? The matrix-chain multiplication problem is a good example of why the order can matter. There, all matrices eventually need to be multiplied, but different orderings require vastly different numbers of operations.

At any given point in time we will have a set of sets of vectors and we are free to choose any two to cross-sum. We remove them from the set, cross-sum them and put the result back into the set. When there is one set left, this is our answer. The regular IP algorithm always uses the previous result as one of the sets (see equation 43).

The analysis here assumes that all of the sets to be cross-summed are parsimonious.

The NCS-SP and RR algorithms have their own criteria for how best to do the cross-sum given two sets of particular sizes. Recall that we let N be the size of the larger set and M the size of the bigger set. The question becomes deciding which two sets to pick, given nothing but their sizes (i.e., we will not know anything about their resulting cross-sum set.) Among the choices are:

- Select the two smallest sets (IP-SS)
- Select the two largest sets (IP-LL)

- Select the smallest and largest set (IP-SL)

If we list a set of sets in size order from smallest to largest, $|A| < |B| < |C| ... < |Z|$, then the parenthesization that IP-SS and IP-LL have are

$$((\ldots((A \oplus B) \oplus C) \oplus \ldots) \oplus Z) \tag{44}$$
$$(A \oplus (\ldots \oplus (X \oplus (Y \oplus Z)) \ldots)) \tag{45}$$

respectively. The somewhat surprising observation is that for the IP-SL it is nearly the same as IP-SS. If we let L be the largest set then the parenthesization for IP-SL is

$$((\ldots(((L \oplus A) \oplus B) \oplus C) \oplus \ldots) \oplus Z) \tag{46}$$

This is a direct result of the fact that the cross-sum of two (parsimonious) sets has to be at least as large as the largest set in the cross-sum.

5.1 Set selection of three sets - total LPs

5.2 Set selection of three sets - total constraints

The first step in a complete understanding of this problem is to look at the simple case

$$A \oplus B \oplus C$$

where $|A| < |B| < |C|$. We want to know which two sets to choose to cross-sum first. Notice that if we choose B and C, the size of the result must be at least size $|C|$ (assuming they are parsimonious), and so the result will be larger than the remaining A set. The same argument holds for choosing A and C first; the result will be bigger than the remaining B set. It is only when we first choose A and B that there will be two possibilities for the size of the result: $|A \oplus B| \leq |C|$ and $|A \oplus B| > |C|$. This is important to keep in mind when doing the analysis using the RR algorithm.

We are now going to abuse notation fairly heavily and use the letters A, B and C and the size of the sets. The context that they are used in will disambiguate whether we are referring to a set or its size.

To do an analysis of a sequence of cross-sums, we have to impose some restrictions upon the result size. In general, we cannot predict the size of the result, so we resort to using a parameter ρ, where

$$U = \rho NM$$

This is a sort of expected number of useless vectors for a cross-sum of N and M vectors. Returning to the case when we select A and B first, we can see that the ordering for the second sum is determined by the relationship of ρAB and C. It will be convenient to define $\alpha = 1 - \rho$, which makes the expected size of the cross-sum of two sets αNM.

With the average case number of constraints for the cross-sum algorithms already worked out (see equations 11, 17 and 30), and with the expected size of the result being αNM, we can write down a formula for the expected number of constraints for each of the three possible choices for finding $A \oplus B \oplus C$. Note that for this analysis we are restricted to $0 < \rho 1 - (AB)^{-\frac{1}{2}}$, since the size of the the cross-sum of these three sets must be at least C, i.e., $\alpha^2 ABC \geq C$. We also divide by ρ to derive some of these formula, so we treat that as a special case. It can be shown that for $\rho = 0$ IP-SS is always the best choice, regardless of the sizes of A, B and C (see appendix J).

It turns out that all three choices have a common term of

$$\rho\alpha ABC - A^2 - \frac{\alpha}{2} ABC$$

So we can ignore this in the comparison, which simplifies the formulas. The IP-SS actually has two cases, depending upon the relationship between αAB and C. If $\alpha AB < C$ we have

$$\mathrm{IPSS}_{\mathrm{ave}}(A,B,C) = AB\left(A + \gamma_1 B + \gamma_5 AB - \alpha^2 AB\right) + \gamma_4 C \tag{47}$$

otherwise, if $\alpha AB \geq C$ we use

$$\mathrm{IPSS}_{\mathrm{ave}}(A,B,C) = AB\left(A + \gamma_1 B + \gamma_2\right) - C^2 + \gamma_3 C \tag{48}$$

The other two are easier cases. For IP-LL we have

$$\mathrm{IPLL}_{\mathrm{ave}}(A,B,C) = BC\left(B + \gamma_1 C + \gamma_2\right) - B^2 + \gamma_3 B \tag{49}$$

and IP-SL

$$\mathrm{IPSL}_{\mathrm{ave}}(A,B,C) = AC\left(A + \gamma_1 C + \gamma_2\right) - B^2 + \gamma_3 B \tag{50}$$

Here we have conveniently defined the following

$$\begin{aligned}
\kappa &= \rho\alpha ABC \\
\gamma_1 &= \frac{\alpha}{2} \\
\gamma_2 &= \frac{2\rho - 1}{2} + \frac{\kappa\alpha^2}{2\rho} \\
\gamma_3 &= \alpha ABC \\
\gamma_4 &= \frac{\kappa\alpha}{2\rho} \\
\gamma_5 &= \frac{2\rho - 1}{2} + \frac{\kappa\alpha}{\rho}
\end{aligned}$$

Note that these constants can probably be simplifed.

So given three sets and an expectation on the percentage of useless vectors in a cross-sum, we could just compute these values and choose the one with the least number of expected constraints. We note that for each of the three choices, we can construct problem sizes where it will have fewer expected constraints than the other two. The question is when will these cases happen?

Until my analysis can quantify this better, here are some observations from an empirical comparison of these three choices:

- IP-SS - Best most often except for the exceptions cited below.
- IP-LL - never best when $\rho \leq 0.5$. The smaller the set sizes, the less likely it is to be good. When set sizes are big it is more likely to be good when ρ is around 0.9. This seems never to be better than IP-SL when $\rho < 0.75$.
- IP-SL - more likely to be best when set sizes are greater than 20 or so and the peaks seems to be $0.7 < \rho < 0.9$, depending upon the size. Where it peaks seems to be a function of the size of the problems. The larger the set sizes, the closer the small-n-large peak is to 0.9

We now move onto the question of how this simple case can be used in the analysis of the case when we have an arbitrary number of sets to choose the cross-sum ordering of.

6 Incremental Pruning (IP) Analysis

We now attempt to analyze an entire Q^a construction using the IP algorithm. Currently, it does not draw on any of the discussion of section 5, but makes some simplifying assumptions and limits itself to the worst-case scenario. We will use $|Q^a| = Q$ and assume that all S_z^a sets have size M.

We note that is all the S_z^a sets are parsimonious, then $|Q^a|$ set must be at least as large as the largest S_z^a set .

The worst case for this situation is when the two sets become as large as possible as quickly as possible. Since we are assuming that all S_z^a sets are of the same size, the largest set can only get as large as Q and smallest set will always be the size of the S_z^a sets (since $Q \geq |S_z^a|$).

For some analysis we assume that $\forall z, |S_z^a| = M$. The first cross sum that is done will be with two sets of size M. The worst case is when the resulting parsimonious representation of this first cross-sum set becomes as large as Q. For this to happen, $Q \leq M^2$. If $Q > M^2$, then the worst case is when the sequence of cross-sum results have sizes: M^2, M^3, ... Q, Q, ... Q. However, even for the $Q > M^2$ case, if we assume the first cross sum does produce a set size of Q, we will simply be considering more constraints as possible. Thus, we can make this assumption and preserve calling it the worst case possibility.

For some of the analyses, we will not assume that the S_z^a sets are all of the same size. This is the more realistic case, though harder to analyze. For these cases we define $|S_i^a| = S_i$.

As mentioned, the first cross-sum takes two sets of size M and produces a set of size Q. Each of the $Z-2$ subsequent cross-sums will take a set of size M and a set of size Q and produce a set of size Q. Note that it is impossible to generate a parsimonious representation of a cross-sum that is larger than Q. The number of useless vectors in the first cross sum is $M^2 - Q$ and for each subsequent cross-sum is $MQ - Q$.

Sometimes we might actually use the intermediate sizes of the cross-sums directly. The notation for this case will be V_i for the size of the result of the $(i-1)^{\text{th}}$ cross-sum. We define $V_1 = S_1$ and $V_Z = Q$. We note that $\forall i, V_i \leq Q$ (except V_1 when S_1 is not parsimonious).

6.1 IP-NCS

6.1.1 IP with NCS - total LPs - worst case

For this case we do not have to impose any restrictions upon the relative sizes of the S_Z^a sets, We also assume that we only find 2 vectors by checking the simplex corners. Because the results of all the cross-sum are bounded above by Q, this analysis is actually a worst case one, since we assume that every cross-sum results in a set of size Q. Thus, an actual problem is likely to need less LPs than this.

The first cross-sum will require $S_1 S_2 - 2$ LPs, and each subsequent LP will require $QS_i - 2$ LPs. Thus, the total number of LPs required is

$$
\begin{aligned}
\text{IPNCS}_{\text{LPs}}(S_z, Q, Z) =& S_1 S_2 - 2 + \sum_{i=3}^{Z} (QS_i - 2) \\
=& S_1 S_2 - 2 + Q \sum_{i=3}^{Z} S_i - 2(Z-2) \\
=& S_1 S_2 + Q \sum_{i=3}^{Z} S_i - 2(Z-1) \qquad (51)
\end{aligned}
$$

If we assume that for all i, $S_i = M$ then the total number of LPs is

$$
\text{IPNCS}_{\text{LPs}}(M, Q, Z) = M^2 + Q(Z-2)M - 2(Z-1) \qquad (52)
$$

6.1.2 IP with NCS - total constraints - best case

6.1.3 IP with NCS - total constraints - worst case

We will begin the the most general case, imposing no simplifications upon the sizes of V_i or S_i. We will then simplify the results by using upper bounds on these. Note that there are only $Z-1$ cross-sums done when there are Z sets.

$$\begin{aligned}
\mathrm{IPNCS_{worst}}(S_z, Q, Z) &= \sum_{i=1}^{Z-1} \mathrm{NCS_{worst}}(V_i, S_{i+1}, V_i S_{i+1} - V_{i+1}) \\
&= \sum_{i=1}^{Z-1} \frac{1}{2}\left[V_i^2 S_{i+1}^2 + V_i S_{i+1} + V_i S_{i+1} - V_{i+1} - (V_i S_{i+1} - V_{i+1})^2 - 6\right] \\
&= \sum_{i=1}^{Z-1} \left[V_i S_{i+1}(V_{i+1}+1) - \frac{1}{2}\left(V_{i+1}^2 + V_{i+1} + 6\right)\right] \qquad (53)
\end{aligned}$$

This equation is a bit too general and hard to do anything with, therefore since all the V_i cannot be larger than Q, we firsat make the worst case assumption that $\forall i, V_i = Q$.

$$\mathrm{IPNCS_{worst}}(S_z, Q, Z) = S_1 S_2(Q+1) + Q(Q+1)\sum_{i=3}^{Z} S_i - \frac{1}{2}(Z-1)(Q^2+Q+6) \qquad (54)$$

We can further simplify this equation by taking the largest $S_i = M$ and assuming that $\forall i, S_i = M$. This yields

$$\begin{aligned}
\mathrm{IPNCS_{worst}}(M, Q, Z) =& \mathrm{NCS_{worst}}(M, M, M^2 - Q) + \sum_{i=2}^{Z-1} \mathrm{NCS_{worst}}(M, Q, MQ - Q) \\
=& M^2(Q+1) + (Z-2)(MQ^2 + MQ) - \frac{1}{2}(Z-1)(Q^2+Q+6) \qquad (55)
\end{aligned}$$

There is another way we could look at the worst case. This is when every vector is useful, i.e., $Q = M^Z$ and $U = 0$ for each cross-sum. We can derive a formula for the number of constraints by noting that cross-sum i will have $\mathrm{NCS_{worst}}(M^i, M))$ constraints (where $U = 0$) Thus, an alternative worst case formula could be

$$\begin{aligned}
\mathrm{IPNCS_{worst}}(M, Q, Z) &= \sum_{i=1}^{Z-1} \frac{1}{2}\left[M^{2i+1} + M^{i+1} - 6\right] \\
&\text{Put intermediate steps here.} \\
&= \frac{1}{2}\left[M^4\left(\frac{M^{2Z-2}-1}{M^2-1}\right) + M^2\left(\frac{M^{Z-1}-1}{M-1}\right) - 6Z + 6\right] \qquad (56)
\end{aligned}$$

We note that plugging in $Q = M^Z$ into equation 55 will yield a higher value than equation 56, since the former makes the assumption that the very first cross sum produces a set of size Q. For $Q = M^Z$, this is impossible and furthermore, every cross sum except the last one will yield sets with less than Q vectors. Therefore, equation 56 is liable to be a tighter upper bound when there are few useless vectors.

6.1.4 IP with NCS - total constraints - average case

6.2 IP-RR-NSP-S

6.2.1 IP with RR-NSP-S - total LPs

$$\begin{aligned}
\mathrm{IPRRNSPS_{LPs}}(S_z, Q, Z) &= \mathrm{LF_{LPs}}(0, 0, S_1 S_2) + \sum_{i=3}^{Z} \mathrm{LF_{LPs}}(0, 0, Q S_i) \\
&= S_1 S_2 + Q \sum_{i=3}^{Z} S_i \qquad (57)
\end{aligned}$$

6.2.2 IP with RR-NSP-S - total constraints - best case

6.2.3 IP with RR-NSP-S - total constraints - worst case

Without much loss of generality we assume $S_1 \geq S_2$ and get

$$\begin{aligned}
\text{IPRRNSPS}_{\text{worst}}(S_z, Q, Z) =& \text{RRNSPS}_{\text{worst}}(S_1, S_2, S_1S_2 - Q) + \sum_{i=3}^{Z} \text{RRNSPS}_{\text{worst}}(Q, S_i, QS_i - Q) \\
& \text{Put in intermediate steps here.} \\
=& S_1S_2^2 + QS_2 + Q\sum_{i=3}^{Z} S_i^2 + (Z-2)Q^2 - \frac{1}{2}(Z-1)Q - \frac{Q^2}{2}\sum_{i=2}^{Z}\frac{1}{S_i} \qquad (58)
\end{aligned}$$

Assuming $\forall i, S_i = M$ we get

$$\text{IPRRNSPS}_{\text{worst}}(S_z, Q, Z) = (Q(Z-2) + M)\left(M^2 + Q\right) - \frac{1}{2}(Z-1)Q\left(1 + \frac{Q}{M}\right)$$

6.2.4 IP with RR-NSP-S - total constraints - average case

6.3 IP-NCS-SP

6.3.1 IP with NCS-SP - total LPs - worst case

We assume that $S_1 \geq S_2$ and $\forall i, Q \geq S_i$, so the first cross-sum will require $S_1S_2 - S_1$ LPs, and each subsequent LP will require $QS_i - Q$ LPs. Thus, the total number of LPs required is

$$\begin{aligned}
\text{IPNCSSP}_{\text{LPs}}(S_z, Q, Z) =& \text{LF}_{\text{LPs}}(S_1, 0, S_1S_2 - S_1) + \sum_{i=3}^{Z} \text{LF}_{\text{LPs}}(Q, 0, S_iQ - Q) \\
=& S_1S_2 - S_1 + \sum_{i=3}^{Z}(QS_i - Q) \\
=& S_1(S_2 - 1) + Q\left(\sum_{i=3}^{Z} S_i - Z + 2\right) \qquad (59)
\end{aligned}$$

If we assume that for all i, $S_i = M$ then the total number of LPs is

$$\text{IPNCSSP}_{\text{LPs}}(M, Q, Z) = (M-1)(M + Q(Z-2)) \qquad (60)$$

6.3.2 IP with NCS-SP - total constraints - best case

6.3.3 IP with NCS-SP - total constraints - worst case

We have to be a bit careful about the worst case here. With regular IP with NCS, the assumption that the very first cross-sum yields a set of size Q is valid for the worst case, since the realtive sizes the two cross-sum sets does not come into play. However, the NCS-SP variation is sensitive to the relative sizes of Q and the S_z^a sets, since it will save $max(Q, S_i)$ LPs for each cross-sum. Here we assume $\forall i, Q \geq S_i$ and note that this still preserves the worst case assumption; since we are assuming that we save Q LPs instead of S_i, when $S_i > Q$, we will be underestimating the number of LP solved.

Without much loss of generality we assume $S_1 \geq S_2$ and get

$$\begin{aligned}\text{IPNCSSP}_{\text{worst}}(S_z, Q, Z) =& \text{NCSSP}_{\text{worst}}(S_1, S_2, S_1S_2 - Q) + \sum_{i=3}^{Z} \text{NCSSP}_{\text{worst}}(Q, S_i, QS_i - Q) \\ & \text{Put in intermediate steps here.} \\ =& (Q+1)\left(S_1S_2 + Q\left[\sum_{i=3}^{Z} S_i - Z + \frac{3}{2}\right]\right) - \frac{1}{2}S_1(S_1+1) \end{aligned} \quad (61)$$

Assuming $\forall i, S_i = M$ we get

$$\text{IPNCSSP}_{\text{worst}}(S_z, Q, Z) = (Q+1)\left(M^2 + Q\left[(Z-2)M - Z + \frac{3}{2}\right]\right) - \frac{1}{2}M(M+1)$$

6.3.4 IP with NCS-SP - total constraints - average case

6.4 IP-RR-S

6.4.1 IP with RR-S - total LPs - worst case

Without loss of generality, assume $S_1 \geq S_2$. Then from equation 22, the first cross-sum will require $\text{RR}_{\text{LPs}}(S_2, S_1) = S_1S_2 - S_2$ LPs. Each subsequent cross-sum will require $\text{RR}_{\text{LPs}}(S_i, Q) = QS_i - S_i$ LPs.

$$\begin{aligned}\text{IPRRS}_{\text{LPs}}(S_z, Q, Z) =& S_1S_2 - S_2 + \sum_{i=3}^{Z}(QS_i - S_i) \\ =& S_1S_2 - S_2 + Q\sum_{i=3}^{Z} S_i - \sum_{i=3}^{Z} S_i \\ =& S_1S_2 + Q\sum_{i=3}^{Z} S_i - \sum_{i=2}^{Z} S_i \end{aligned} \quad (62)$$

If we assume that for all i, $S_i = M$ then the total number of LPs is

$$\text{IPRRS}_{\text{LPs}}(M, Q, Z) = M^2 + Q(Z-2)M - (Z-1)M \quad (63)$$

6.4.2 IP with RR-S - total constraints - best case

6.4.3 IP with RR-S - total constraints - worst case

Worst case total constraints for IP with RR-S is a bit problematic for a number of reasons. It requires that we have a parsimonious representation for the region sets and most other algorithms don't make this restriction. Thus, the cost of making the region set parsimonious should be accounted for somewhere. However, if we are ensuring that the S_z^a sets are parsimonious, then the situation is not so bad, because while making the S_z^a sets parsimonious we generate the witness points we need, so we just have to save them.

We proceed much the same as we did in section 6.1.3, starting with a general case and then making simplifying assumptions to arrive at a less complex equation. We asume that $S_1 \geq S_2$.

$$\begin{aligned}\text{IPRRS}_{\text{worst}}(S_z, Q, Z) =& \sum_{i=1}^{Z-1} \text{RR}_{\text{worst}}(S_{i+1}, V_i, V_iS_{i+1} - V_{i+1}) \\ & \text{Put in intermediate steps.} \\ =& \sum_{i=1}^{Z-1}\left[S_{i+1}^2V_i - S_{i+1}^2 + V_iV_{i+1} - \frac{1}{2}\left(V_{i+1} + \frac{V_{i+1}^2}{S_{i+1}}\right)\right] \end{aligned} \quad (64)$$

This equation is a bit too general and hard to do anything with, therefore since all the V_i cannot be larger than Q, we firsat make the worst case assumption that $\forall i, V_i = Q$.

$$\begin{aligned}\text{IPRRS}_{\text{worst}}(S_z, Q, Z) =& S_1 S_2 - S_2^2 + SQ + (Q-1)\sum_{i=1}^{Z-1} S_{i+1}^2 + (Z-2)Q^2 \\ & - \frac{1}{2}(Z-1)Q - \frac{1}{2}Q^2 \sum_{i=1}^{Z-1} \frac{1}{S_{i+1}} \qquad (65)\end{aligned}$$

We now further simplify and assume $\forall i, S_i = M$. We can plug this into equation 65 or use the following simplified derivation.

Using the expression for the worst case of RR (equation 25), IP with RR-S worst case total constraints becomes

$$\begin{aligned}\text{IPRRS}_{\text{worst}}(M, Q, Z) =& \text{RR}_{\text{worst}}(M, M, M^2 - Q) + \sum_{i=2}^{Z-1} \text{RR}_{\text{worst}}(M, Q, QM - Q) \\ =& M^3 - M^2 + \frac{1}{2}\left[M^3 - M^2 + M^2 - Q - \frac{(M^2-Q)^2}{M}\right] \\ & + \sum_{i=2}^{Z-1}\left(QM^2 - M^2 + \frac{1}{2}\left[Q^2 M - QM + QM - Q - \frac{(QM-Q)^2}{M}\right]\right)\end{aligned}$$

Put intermediate steps here.

$$= M^3 - \frac{1}{2}(Z-1)\left(2M^2 + Q + \frac{Q^2}{M}\right) + (Z-2)(QM^2 + Q^2) + MQ \qquad (66)$$

This derivation always uses M as the size of the region set, thus assuming that $M \leq Q$. If $Q < M$, then RR-S should switch the sets around. However, section 4.2 shows that for nearly every case RR-S has fewer total constraints than RR-L, so equation 66 is still liable to be a valid worst case formula. To get a slightly tighter bound for the case when $Q < M$, we can simply switch the Q and M around in the summation to get

$$\text{IPRRS}_{\text{worst}}(M, Q, Z) = \text{RR}_{\text{worst}}(M, M, M^2 - Q) + \sum_{i=2}^{Z-1} \text{RR}_{\text{worst}}(Q, M, QM - Q)$$

Put intermediate steps here.

$$\begin{aligned}=& M^3 - M^2 - Q\left(Z - \frac{3}{2}\right) + (Z-1)MQ \\ & + (Z-2)(MQ^2 - Q^2) - \frac{Q^2}{2M} \qquad (67)\end{aligned}$$

6.4.4 IP with RR-S - total constraints - average case

7 Incremental pruning comparisons

7.1 IP-NCS vs. IP-RR-NSP-S

7.1.1 IP-NCS vs. IP-RR-NSP-S - total LPs

Equations 51 and 57 show that IP-NCS does $2(Z-1)$ fewer LPs than IP-RR-NSP-S.

7.2 IP-NCS vs. IP-RR-NSP-S - total constraints - worst case

From equations 54 and 58 we get the asymptotic complexities of $O(Q^2 \sum_{i=3}^{Z} S_i)$, $O(Q \sum_{i=3}^{Z} S_i^2 + Q^2 Z)$ for IP-NCS and IP-RR-NSP-S respectively. When $\forall i, S_i = \Theta(Q)$ these equivalent, but otherwise IP-RR-NSP-S has lower complexity. This follows from the analysis of section 4.3.3 which showed, for a single cross-sum, that NCS was only better when $N = M$ and a few other cases.

Empirical observation: IP-RR-NSP-S *is never worse for the* $3,822,588$ *cases of* $4 \leq M \leq 200$, $3 \leq Z \leq 200$, $M + 1 \leq Q \leq 200$. *I need to do the analysis to say something more general about it.*

7.3 IP-NCS-SP vs. IP-RR-S

7.3.1 IP-NCS-SP vs. IP-RR-S - total LPs

Equations 59 and 62 show

7.4 IP-NCS-SP vs. IP-RR-S - total constraints - worst case

8 Witness Analysis

We will want to compare variations of IP with the previous best exact POMDP algorithm, Witness. To do this we first present a brief analysis of the Witness algorithm. We make no attempt to explain this algorithm at this time, but do note that there are differing ways that the algorithm can be initialized.

The first method simply adds an arbitrary item to the agenda and finds all Q witness points in the same manner. Another method, and the one we assume for the analysis, uses an arbitrary point to initialize the Q^a set and adds all of its *neighbors* to the agenda. This latter method only needs to uncover $Q - 1$ witness points. Other options are variations on using the simplex corners and using previously saved witness points. The initialization step assumption affects the analysis to varying degrees.

We only focus on the complexity of constructing a single Q^a set, and we denote the size as $|Q^a| = Q$. We assume that the size of each S_z^a set is M.

How many constraints are in each witness LP depends upon how quickly we uncover the vectors in Q^a. At any given time, the number of constraints in the witness LP will be one more than the current size of Q^a set we are building up. The faster we find witness points, the larger the set grows and larger the LPs get. The worst case is when the first $Q - 1$ LPs all discover witness points. Then all the agenda items will have the largest possible LPs when they are removed. The absolute best case is when we always find the witness points with the very last agenda item.

8.1 Witness - total LPs

The witness algorithm sets up an LP to either remove an item from the agenda or to find a witness point. In the general case there will be $(S_z - 1)$ agenda items for each observation and vector in Q^a. Thus, there will be a total of $Q \sum_{i=1}^{Z} (S_i - 1)$ agenda items to be processed.

$$\begin{aligned} \text{WITNESS}_{\text{LPs}}(S_z, Q, Z) =& Q \sum_{i=1}^{Z} (S_i - 1) + Q - 1 \\ =& Q \left(\sum_{i=1}^{Z} S_i - Z \right) + Q - 1 \end{aligned} \tag{68}$$

Assuming $\forall i, S_i = M$, there will be $Z(M - 1)$ items added to the agenda for each vector in Q^a, so there will be $QZ(M - 1)$ items in the agenda that will need to be removed. Thus the total number of LPs done in constructing Q^a

$$\text{WITNESS}_{\text{LPs}}(M, Q, Z) = QZ(M - 1) + Q - 1 \tag{69}$$

8.2 Witness - total constraints - best case

The best case analysis is convenient, because if we can show that the worst case of an IP variation is better than this, we can say something very strong about that variation in comparison to the Witness algorithm.

We start with one vector in Q^a and add its $\sum_{i=1}^{Z}(S_i - 1)$ *neighbors* to the agenda. The LPs stay as small as possible as long as we don't find a witness point (only 2 constraints at first). Thus we want to find the witness point with the very last agenda item. We note that after we find this witness point, we will still need an LP to remove this item from the list. After this second vector is added to Q^a, we add another $\sum_{i=1}^{Z}(S_i - 1)$ items to the agenda and want to find the next witness point with the last agenda item again.

$$\begin{aligned}
\mathrm{WITNESS}_{\mathrm{best}}(M,Q,Z) =& 2\sum_{i=1}^{Z}(S_i - 1) + \sum_{i=3}^{Q+1}(\sum_{i=1}^{Z}(S_i - 1) + 1)i \\
& \text{Add intermediate steps.} \\
=& \frac{1}{2}\left(\sum_{i=1}^{Z} S_i - Z + 1\right)(Q+1)(Q+2) - \sum_{i=1}^{Z} S_i + Z - 3 \qquad (70)
\end{aligned}$$

Assuming $\forall i, S_i = M$

$$\begin{aligned}
\mathrm{WITNESS}_{\mathrm{best}}(M,Q,Z) &= 2Z(M-1) + \sum_{i=3}^{Q+1}(Z(M-1)+1)i \\
&= 2Z(M-1) + (Z(M-1)+1)\sum_{i=3}^{Q+1} i \\
&= 2Z(M-1) + (Z(M-1)+1)(\sum_{i=1}^{Q+1} i - 3) \\
&= 2Z(M-1) + (Z(M-1)+1)(\frac{1}{2}(Q+1)(Q+2) - 3) \\
&= 2Z(M-1) + 2 - 2 + (Z(M-1)+1)(\frac{1}{2}(Q+1)(Q+2) - 3) \\
&= (Z(M-1)+1)(\frac{1}{2}(Q+1)(Q+2) - 1) - 2 \qquad (71)
\end{aligned}$$

8.3 Witness - total constraints - worst case

The worst case total constraints for witness is when we find all the witness points before we remove anything from the agenda.

$$\begin{aligned}
\mathrm{WITNESS}_{\mathrm{worst}}(M,Q,Z) &= \sum_{i=1}^{Q-1}(i+1) + \sum_{i=Q}^{Q(\sum_{j=1}^{Z} S_j - Z)+Q-1}(Q+1) \\
&= \sum_{i=1}^{Q} i + \sum_{i=1}^{Q(\sum_{j=1}^{Z} S_j - Z)}(Q+1) - 1 \\
&= Q(Q+1)\left(\sum_{i=1}^{Z} S_i - Z - \frac{1}{2}\right) \qquad (72)
\end{aligned}$$

If we assume $\forall i, S_i = M$ we get

$$
\begin{aligned}
\text{WITNESS}_{\text{worst}}(M,Q,Z) &= \sum_{i=1}^{Q-1}(i+1) + \sum_{i=Q}^{QZ(M-1)+Q-1}(Q+1) \\
&= \sum_{i=1}^{Q} i + \sum_{i=1}^{QZ(M-1)}(Q+1) - 1 \\
&= \frac{1}{2}Q(Q+1)QZ(M-1)(Q+1) - 1 \\
&= Q(Q+1)(Z(M-1) - \frac{1}{2}) \qquad (73)
\end{aligned}
$$

8.4 Witness - total constraints - average case

This case isn't that useful, so we adopt a simple interpretation of the average case. We assume that we do the same number of LPs for all possible sizes of LPs. The LP sizes vary from 2 to $Q+1$ and there are a total of $QZ(M-1)+Q-1$ LPs. The average LP size is $(Q+3)/2$ and thus

$$
\begin{aligned}
\text{WITNESS}_{\text{ave}}(M,Q,Z) &= \frac{1}{2}(Q+3)(QZ(M-1)+Q-1) \\
&= \frac{1}{2}(Q+3)(Q(Z(M-1)+1)-1) \qquad (74)
\end{aligned}
$$

It is conjectured that the actual average case would have more constraints than this, since we would expect witness points are more likely to be uncovered early on in processing agenda items than later.

9 Incremental Pruning vs. Witness

9.1 IP-NCS vs. Witness - total LPs

The analysis for witness (equation 68) is the number of LPs that must always be done. Equation 51 for IP-NCS used a worst case analysis. If we assume that all S_z^a are parsimonious to start with, then $Q \geq \max_i S_i$.

Empirical observation: IP-NCS *is better when* $Q \geq M$ *and* $Z \leq M$. *Witness is better when* $Q \leq M-3$.

Using equations 68 and 51 we can derive the following inequality

$$
\begin{aligned}
Z(Q-2) &\leq Q(S_1+S_2+1) - S_1S_2 - 3 \qquad (75) \\
Z &\leq \frac{Q(S_1+S_2+1) - S_1S_2 - 3}{Q-2} \qquad (76)
\end{aligned}
$$

When this inequality is satisfied, then witness will never do fewer LPs than IP-NCS. If we restrict ourselves to the case where $\forall i, Q \geq S_i$, then the simple constraint $Z \leq \max(S_1, S_2)$ will be enough to specify when IP-NCS will require less LPs than witness. This follows from arguments that stem from the following rewrite of right-hand side of equation 76

$$
S_1 + S_2 + \frac{2S_1}{Q-2} + \frac{2S_2}{Q-2} - \frac{S_1S_2}{Q-2} - \frac{1}{Q-2} \qquad (77)
$$

The last term will always be less than one and is a negligible contribution to the right-hand side. Under the assumption that $\forall i, Q \geq S_i$ the fifth term will contribute at most $-\min(S_1, S_2)$ LPs. The third and fourth terms only serve to increase the right-hand side. We are essentially left with $Z \leq S_1 + S_2 - \min(S_1, S_2)$ which shows that for $Z \leq \max(S_1, S_2)$ IP-NCS is at least as good as witness.

Note that we divided by $Q-2$ to derive equation 76. However, we always assume that $Q > 1$ and we can show that equation 75 is satisfied for $Q = 2$.

Also note that because the IP-NCS analysis was worst case, the situations where it is better will probably be broader than what is specified here.

9.2 IP-NCS vs. Witness - total constraints - worst case

Using equations 53 and 72 we can derive the following inequality

Put derivation here.

$$Z \leq 2(S_1 + S_2) - \frac{2S_1S_2}{Q} - 2 \tag{78}$$

which, when satisfied ensures that IP-NCS has no more constraints than witness. When we assume $Q \geq \max(S_1, S_2)$ the second term on the right-hand side contributes at most $-2\min(S_1, S_2)$ constraints. This leaves us with the inequality $Z \leq 2\max(S_1, S_2) - 2$.

Switching the inequality sign in equation 78 we get an expression that, when satisfied, will ensure that witness does no more LPs than IP-NCS.

Empirically we found this inequality to be satisfied when $M \geq 3$ and $Z \geq 4M - 1$. So it should be easy to show this by doing the algebra.

For the cases in between the cases where we show the two variations to be superior, we would like to quantify which will be better. This will require some extra work.

9.3 IP-RR-NSP-S vs. Witness - total LPs

In section 7.1.1 we showed that IP-RR-NSP-S does two more LPs than IP-NCS. Thus, the comparison shown in section 9.1 is nearly identical to this case with the exception of the constant factor 2 which has to be accounted for.

Need to show the derivation and maybe add some more discussion here, but it is mostly the same as section 9.1, so we'll leave it an an exercise for the reader.

9.4 IP-RR-NSP-S vs. Witness - total constraints - worst case

Since IP-NCS was not a clear winner over witness, the results of section 7.2 do not let us say anything very strong about the relationship between IP-RR-NSP-SP and witness. Therefore, we must directly compare them.

Asymptotically, IP-RR-NSP-S is $O(Q\sum_{i=3}^{Z} S_i^2 + Q^2 Z)$ and witness is $O(Q^2\sum_{i=1}^{Z} S_i)$.

Not sure if we can really make any asymptotic claims here.

Using equations 58 and 72 we can derive an inequality to determine when one algorithm has fewer constraints than the other.

We need to analyze this inequality better to see what we can prove about it. Empirically, when $M > 2$, and $Q > M + 2$ IP-RR-NSP-S *never has more constraints than witness. Thus, there are many more cases where* IP-RR-NSP-S *is better than witness than there were for* IP-NCS. *We especially need to characterize the cases where witness is better.*

9.5 IP-NCS-SP vs. Witness - total LPs

9.6 IP-NCS-SP vs. Witness - total constraints - worst case

9.7 IP-RR-S vs. Witness - total LPs

Using equations 62 and 68 we can derive the inequality

$$S_1S_2 + Q\sum_{i=3}^{Z} S_i - \sum_{i=3}^{Z} S_i \leq Q\left(\sum_{i=1}^{Z} S_i - Z\right) + Q - 1$$

Put intermediate steps here.

$$Z \leq \frac{1}{Q}\left(\sum_{i=2}^{Z} S_i - 1\right) + S_1 + S_2 + 1 \tag{79}$$

which, when satisfied, ensures that IP-RR-S does not more LPs than witness. If $\sum_{i=2}^{Z} S_i < Q$ then $Z \leq S_1 + S_2$ will ensure than IP-RR-S is better. When $\sum_{i=2}^{Z} S_i \geq Q$ this gets even less restrictive.

I did a poor job quantifying the cases here. I should go back over this when I put in the intermediate steps.

Assuming that $\forall i, S_i = M$ we get

$$Z \leq 2M + 1 + \frac{M^2 - 1}{Q - M} \tag{80}$$

which shows that IP-RR-S is best when $Z \leq 2M + 1$.

9.8 IP-RR-S vs. Witness - total constraints - worst case

Nothing but empirical results here so far. But it pretty much looks like IP-RR-S *is always better when* $M > 2$. *For the* $3,920,499$ *cases where* $3 \leq M \leq 200$, $2 \leq Z \leq 200$ *and* $M \leq Q \leq 200$ IP-RR-S *is better.*

10 Effects of Domination Checks in NCS and RR

We have ignored the effects of the domination checks in the algorithms up to this point. However, doing the domination check is the dominant factor in saving time when solving real problems.

Just like trying to know the resulting size of a cross-sum operation, knowing how many vectors the domination check will eliminate is not possible without actually doing the operation. We will therefore quantize the savings similar to what we did earlier. For a given set of size N, assume that the number of vectors purged through the domination check is $(1 - \lambda)N$, which makes the resulting set size λN.

Our first task is to define where the NCS-SP and RR algorithms perform their domination checks. We note that there is a filtering that happens in the construction of the S_z^a sets, but we ignore this, since we can just define the size of the S_z^a set to be the post-filtering size.

10.1 Domination checks in NCS-SP

In the NCS-SP algorithm, the domination check is done after the full cross-sum, just prior to Lark filtering. The resulting size that we will actually need to filter is then λNM. Equations 2, 4 and 5 still apply, except now $Y = \lambda NM - U - N$. This makes the average case total number of constraints in NCS-SP with domination checks

$$\begin{aligned} \mathrm{NCSSPDOM}(N, M, U) &= \mathrm{LF}_{\mathrm{ave}}(N, U, \lambda NM - U - N) \\ &= \frac{1}{2}\left(\lambda^2 N^2 M^2 - N^2 - \lambda UNM + \lambda NM + UN + U - N\right) \end{aligned} \tag{81}$$

Note that it is no longer valid to make the substitution $U = \rho NM$, when looking at the expected total constraint case; we will have to use $U = \rho\lambda NM$ instead. Also note that we are constrained to $1/M \leq \lambda \leq 1$.

10.2 Domination checks in RR

At first it may not appear that domination checks can be used in the RR algorithm. Since we start with two parsimonious sets and never consider the full cross-sum set, how could we possible do domination checks? The answer is that we will actually perform the full cross-sum, before doing the restricted region LPs.

It may seem that doing the full cross-sum defeats the purpose of doing the restricted region technique, but actually doing the cross-sum is a cheap (in time) calculation, assuming we have the space to store all the vectors. The real time consumption is filtering this set down to its parsimonious representation. By doing the full cross-sum and performing domination checks on the full set, we can eliminate vectors that we will have to consider when doing the restricted regions. Since domination checks typically remove a lot of vetcors, this savings is well worth the small amount of extra work, when space permits.

The main drawback is that it complicates the algorithm a bit. First, when we perform the full cross-sum, we must mark each new vector with which region set vector created it. The other complication is that we will not actually be setting up LPs on the N vectors being cross-summed; will will actually be setting up region LPs on the subset of the cross-sum set that were derived from the current region set vector. *This is probably really confusing, so I should introduce some notation to keep things clear.*

Note that a slight savings could be had by removing the M vectors from NM that we could get with the points saved for the region sets. This would reduce the complexity of the domination check. However, this savings is likely to be miniscule compared to the other work that needs to be done.

This domination check in RR also complicates the analysis. Before, we introduced the λ parameter as a measure of how much filtering the domination check accomplishes. For the NCS-SP algorithm this was easy to reason about, since we were just doing Lark filtering on a set of size λNM, but here it would seem to require knowing how the dominated vectors are distributed among the different region set vectors. It will turn out that this doesn't matter as the following derivation shows.

Let λ be defined as before and let δ_i be the number of vectors removed by domination checks on region i. Then the number of LPs for region i will be $N - 1 - \delta_i$. Note that $\sum_{i=1}^{M} \delta_i$ is just the total number of vectors removed by domination checks and thus $\sum_{i=1}^{M} \delta_i = (1 - \lambda)NM$.

Finish this derivation.

A possible interesting result is that the number of constraints saved by domination cheks is different in NCS-SP and RR. This results because RR is better, and it seems to be such that the more vectors it has, the more constraints it saves. Therefore, it would seem that by reducing the number of vectors with a domination check, would narrow the gap between NCS-SP and RR. However, this gap should only go to zero where NCS-SP and RR have equivalent number of constraints (which is to say in uninteresting cases.) *I need to work through these formulas more carefully though to exactly quantify this.*

10.3 Domination checks in Witness

11 RR analysis when $\sigma < 1/2$

All of the analysis on RR assumed that the average case had $\sigma = 1/2$. Since we have shown that lower values for σ only reduce the number of constraints and that RR is superior to NCS-SP with $\sigma = 1/2$, a more detailed analysis is not necessary at this time.

12 Adding LP set-up overhead costs

The total number of constraints criteria was convenient for the initial analysis, but insuffcient for being indicative of a true implementation. It ignores the time required to set up an LP and the efficiency gains of adding more constraints to an LP.

For instance: An algorithm that uses 100 LPs with 1 constraint each could be inferior to an algorithm that used 10 LPs with 10 constraints. Aside from having to set up 90 more LPs, the running time of LP solution procedures does not scale linearly with the number of constraints. The latter problem is too complex to deal with directly, but we would like to capture some notion of of adding in the set-up time.

To do this we can introduce an LP set-up cost . To keep things compatible, we will measure this cost in constraints. This cost could be guessed or measured empircially, then the formulas derived could be compared with whatever particular value you have for the cost.

The number of LPs for NCS-SP is $U + Y$ regardless of whether we are using worst, average or best case analysis. We simply need to check every vector that isn't in the initial set.

In the RR algorithm using the largest set as the region set, for each region of the M regions, we will require $N - 1$ LPs, so the total number of LPs required is $M(N - 1)$. This is larger than the number of LPs done in NCS-SP since $U + Y = NM - N$ and $N \geq M$. Thus RR does $N - M$ more LPs.

Note that if we choose the larger set RR-L, as the region set, then the total number of LPs in RR will be $N(M - 1)$, which is exactly the same as the number for NCS-SP. If we define the overhead for each LP as κ, then for a given overhead we can analyze when RR-L and or NCS-SP is better than RR-S.If either does less than $\kappa(N - M)$ more constraints, then RR would be inferior.

Add the analysis here.

13 Getting witness points for NCS-SP and RR

13.1 Note on NCS

There are instances where NCS could be better than either NCS-SP or RR. Our analysis has ignore the cost of acquiring the initial points that both NCS-SP and RR use. If this cost exceeds the difference between the algorithms, then NCS would be the preferred algorithm. We could go through the analysis if we really wanted to, but for now are content with calling attention to this fact. We suspect that NCS will only win when the problems are on the small side and so the savings are not that important.

13.2 Using witness points in Witness

Talk about seeding with simplex corner and/or saved witness points.

14 Hidden Costs

14.1 Enumerating vectors

A Lark Filter - average case formula proof

B Proof that RR-S better than RR-L - worst case

C Proof that RR-S better than RR-L - average case

D RR-L savings when $S = 2$ - average case

E Proof of $x_i = y/n$

Note: I am absolutely sure that this can be shown. Until I put it here, be a bit skeptical.

F Proof that RR-S better than RR-L - worst case

This is true empirically and I am fairly confident that I can prove this.

G Proof that RR-NSP-S better than NCS - worst case

H Proof that RR-S better than NCS-SP - worst case

I Proof that RR-S better than NCS-SP - average case

J IP set ordering when $\rho = 0$

Note: I am not really sure that this can be shown. Until I put it here, be skeptical.