Generalized Difference Constraints

Piotr Wojciechowski
LDCSEE,
West Virginia University,
Morgantown, WV
{pwojciec@mix.wvu.edu}

December 15, 2014

1 Summary

In studying the generalized difference constraint closure we want a method of precomutation that makes any call to all two variable optimization functions computeable quickly. To do this we need to find all the breakpoints when the system is projected onto the x_i, x_j plane for each pair i, j. Once we have these breakpoints we can quickly find the maximum an minimum values obtained by any optimization function using only the variables x_i and x_j. This proceeds as follows:

Algorithm 1 Pre-processing Algorithm for generalized difference constraint closure

```
GEN-DIF-CLOS (system G of generalized difference constraints)
1: for (Each pair $i$ and $j$) do
2:     Use Fourier-Motzkin Elimination to eliminate all variables in $G$ except $x_i$ and $x_j$.
3:     Store the resultant non-redundant constraints in order of increasing slope.
4: end for
```

Thus, the running time of this algorithm depends on the number of breakpoints between each pair of variables. In the general case we were unable to determine if there are only polynomialy many of these breakpoints or not. However, we have determined several cases where the number of breakpoints is polynomial in the number of variables.

1. If the coefficients use a fixed number, p,of prime factors, then we have a limit of $q(n)^{2 \cdot p}$ possible x_i, x_j constraints that can be generated from the system, where $q(n)$ is some polynomial function of n.
2. When the coefficients are limited by some fixed constant, k, we have obtained a similar result.

There are still further ares to research about this problem.

1. We still do not know if either of this problem is Fixed Parameter Tractable with respect to either p or k.
2. We do not know the impact on accuracy caused by limiting the number of x_i, x_j breakpoints stored. This might lead us to a faster approximation algorithm.

2 December 15, 2014

Let us consider limiting the maximum number of x_i, x_j breakpoints to 8. At each variable elimination we will prune the number of constraints for each pair of variables. Thus at each elimination phase the x_i, x_j breakpoints will look something like Figure 1.

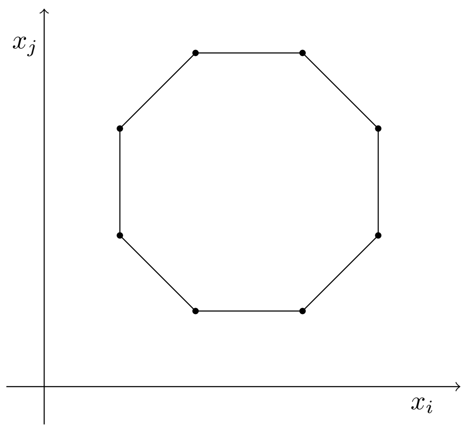


Figure 1: Restricted breakpoints

Since all these points are along the exterior of the convex hull we have that all feasible values of (x_i, x_j) must lie within the shape shown in Figure 2.

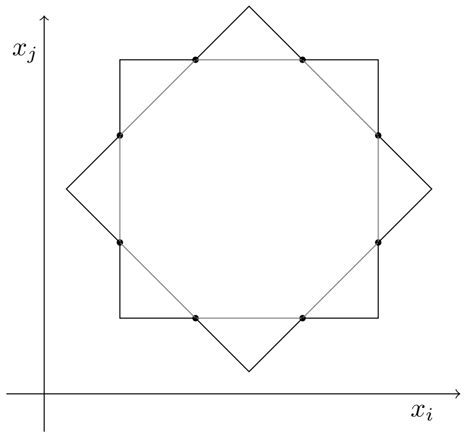


Figure 2: Limit on feasible solutions

This is because if a valid solution is outside those boundaries then, as shown by the red lines in Figure 3, the set of solutions is no longer convex.

However, this does not give us a good approximation. The distance between the actual and computed optimum values will be bounded by the size of the solution space.

3 December 8, 2014

As before we can consider the number of prime factors p in the coefficients of the constraints. We know that $p \leq k$. Since p can be number of primes less than k. By the Prime Number Theorem, as k grows p approaches $\frac{k}{\ln k}$. From the constraints

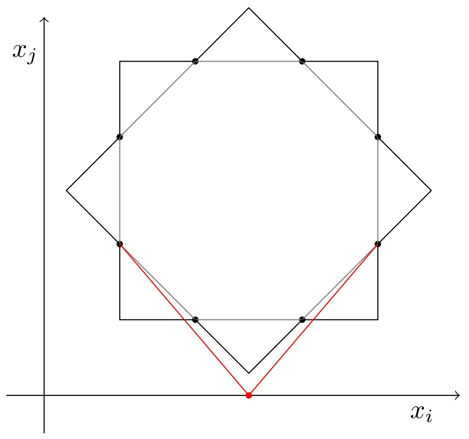


Figure 3: Infeasible solution

$a_i \cdot x_i - b_j \cdot x_j \leq c_{ij}$ and $a_j \cdot x_j - b_l \cdot x_l \leq c_{jl}$ we can derive the constraint $a_i \cdot a_j \cdot x_i - b_j \cdot b_l \cdot x_l \leq a_j \cdot c_{ij} + b_j \cdot c_{jl}$.

Consider the derived constraint $a_1 \cdot x_1 - b_n \cdot x_n \leq c_{1n}$ and let y denote the number of constraints used to derive this constraint. We know that both a_1 and b_n use at most p primes and are at most k^y. When we consider the exponents in the prime factorization of a_1, we know that these exponents add up to at most $y \cdot \log_2 n$. We also have that the more prime factors are involved in this factorization the lower this sum will be.

Grade: C.

4 December 1, 2014

Thanksgiving Break

5 November 24, 2014

Now, instead of limiting the number of prime factors p, we will limit the coefficients themselves. Note that if we have a maximum value, k, for our coefficient then we still limit the total number of prime factors. This limit also limits the number of constraints between each pair of variables and the exponents that the prime factors can have in each coefficient. We know that there are at most $2 \cdot k^2$ constraints involving each pair of variables. We also know that for each coefficient the sum of the exponents of the prime factors is at most $\log_2 k$. We need to see if these additional limits on the problem make it FPT with respect to the parameter k.

Grade: B.

6 November 17, 2014

There does not seem to be a way to limit the number of possible exponents of each prime to be lower than $O(n)$. Thus for each fixed p there will be at least $(2 \cdot n)^p$ possible x_1, x_n constraints generated. Thus under the running time will not be FPT without further restrictions on the coefficients or the choice of a different parameter.

Grade: C.

7 November 10, 2014

To show that Generalized difference constraint closure is FPT we need to change the current running time of $n^{O(p)}$ to $f(p) \cdot n^{O(1)}$ for some computeable function f. First we will look each prime p_i used individually and try to determine the number of possible exponents more accurately that $n^{O(1)}$.

We cannot reduce the maximum number of constraints by only considering the possible combinations of exponents among the prime factors. We only limit the number of primes that can occur as factors in the coefficients and not the number of constraints. Thus within the coefficients themselves we can already have various combinations of prime factors. This means that if, in a generated constraint, we know the exponent of one prime we cannot exclude any values for the exponents of the other primes.

Grade: C.

8 November 3, 2014

We now have a parameter, p the number of prime factors, such that for each fixed p the generalized difference constraint closure problem can be solved in time $n^{O(1)}$. Thus it may be possible to find an FPT algorithm for this problem. The main issue that we currently have a running time of $n^{O(p)}$ instead of $f(p) \cdot n^{O(1))}$. We currently assign the same range of possible exponents, 0 to $q(n)$, to each of the p primes used in the system, if we can get a more accurate exponent for each of the primes then we might be able to get the running time in the desired form.

To aid in this I have read up on the various FPT techniques covered in the slides for the FPT school to see if they can be applied to this problem.

Grade: B.

9 October 27, 2014

For each fixed number of primes used, p, we generate at most $q(n)^{2 \cdot p}$ constraints. Each generated constraint is equivalent to a constraint of the form $x_i - \left(\prod \frac{b_{kl}}{a_{kl}}\right) \cdot x_j \geq c'_{ij}$ which has $q(n)^p$ possible numerators and $q(n)^p$ possible denominators. Thus for each fixed number of primes the Fourier-Motzkin elimination procedure runs in time polynomial in both n and m. Thus, for each fixed p, we have a polynomial algorithm for determining the closure of a set of generalized difference constraints. This proceeds as follows:

Algorithm 2 Pre-processing Algorithm for generalized difference constraint closure

```
GEN-DIF-CLOS (system G of generalized difference constraints)
1: for (Each pair $i$ and $j$) do
2:     Use Fourier-Motzkin Elimination to eliminate all variables in $G$ except $x_i$ and $x_j$.
3:     Store the resultant non-redundant constraints in order of increasing slope.
4: end for
```

Theorem 9.1 *Each $\prod a_{kl}$ and $\prod b_{kl}$ generated by the reduction process has size polynomial in n.*

Proof: We know that each extreme point of the system must have size polynomial in n. We also have that each x_i, x_j breakpoint generated corresponds to the x_i and x_j coordinates of at least one of these extreme points. Thus the size of the generated break points must also be polynomial in n. Since the size of a breakpoint depends on the size of $\prod \frac{b_{kl}}{a_{kl}}$ and c'_{ij} for two generated constraints we must have that the size of all these values are also polynomial in n. □

This means that for each prime factor of $\prod a_{kl}$ and $\prod b_{kl}$ must have its exponent be polynomial in n. Since otherwise either $\prod a_{kl}$ or $\prod b_{kl}$ would be too large. Let $q(n)$ be a polynomial in n that represents the largest exponent of any prime factor. Since we have a fixed number p of primes there are at most $q(n)^{2 \cdot p}$ non-redundant constraints. This means that there are at most $q(n)^{2 \cdot p}$ x_i, x_j breakpoints for each pair x_i and x_j. Thus, for each fixed number of primes p, both the

Fourier-Motzkin elimination procedure and the closure algorithm run in time polynomial in n. However this running time is on the order of $q(n)^{2 \cdot p}$ for some unknown polynomial $q(n)$.

Grade: A.

10 October 20, 2014

If a fixed number of primes are used in total among the decompositions of each a_{ij} then we do have a polynomial algorithm. We have that all the derivable constraints are also of the form $x_i - a'_{ij} \cdot x_j \geq c_{ij}$. Each prime appearing in the decomposition of a'_{ij} must appear in the one of the original a_{ij}s since a'_{ij} is the product of one or more of these values. Thus if the number of primes in the decompositions is p then after the elimination procedure we generate at most $(m \cdot n)^p$ possible a'_{ij}s thus limiting the number of constraints generated an similarly the number of breakpoints. One result of this is that if we have a fixed maximum value for all a_{ij}s then the problem is polynomial in n.

This also works in the case of constraints of the form $a_{ij} \cdot x_i - b_{ij} \cdot x_j \geq c_{ij}$ where both a and b have some maximum size independent of n. Note that all constraints generated by elimination will equivalent to constraints of the form $x_i - \left(\prod \frac{b_{kl}}{a_{kl}}\right) \cdot x_j \geq c'_{ij}$. Both the numerator and denominator of $\prod \frac{b_{kl}}{a_{kl}}$ are limited in the same way that a'_{ij} was previously. Thus resulting in at most polynomially many constraints generated.

Grade: B.

11 October 13, 2014

Let us try to find restricted forms of generalized difference constraints that result in a polynomial number of break points. Let us first focus on the case where each constraint is of the form $x_i - a_{ij} \cdot x_j \geq c_{ij}$ were a is polynomial in n. Note that this is not the same as the size of a being polynomial in n as this still allows a to be singly exponential in n. One advantage of this particular case is that it allows us to in polynomial time and space construct an equivalent set of horn constraints. Each x_j is replaced with $\max_i a_{ij}$ variables and each constraint of the form $x_i - a_{ij} \cdot x_j \geq c_{ij}$ becomes a horn constraint with $(1 + a_{ij})$ variables. Another advantage of this form is that it limits the number of primes in the decomposition of each a_{ij}.

Grade: C.

12 October 6, 2014

Fell ill, No Progress.

13 September 29, 2014

To obtain the desired distribution of break points we need to ensure that $\min \frac{M_i}{M_j} \geq \frac{a_n}{a_1}$. This will ensure that, even if two sets of breakpoints are generated using the same number of variable eliminations (and thus the coefficients of x_n have the same number of M variables) the sets of breakpoints associated with particular sets of variable eliminations will not intersect. We also want that no tow variable eliminations produce the same set of breakpoints. Thus we can associate each variable with a unique prime. Thus one possible value for M_i is $p_i \cdot \lceil \frac{a_n}{a_1} \rceil$ this will have the desired effect on the breakpoint distribution.

Thus for each subset S of the variables, and for each $1 \leq k \leq n$ we generate the following x_1, x_n constraint once all the variables in S are eliminated.

$$x_1 - a_k \cdot \prod_{i \in S} \left(p_i \cdot \frac{a_n}{a_1}\right) \cdot x_n \geq \sum_{i \in S} \left(c_i \cdot \prod_{j \in S, j \leq i} \left(p_j \cdot \frac{a_n}{a_1}\right)\right)$$

Thus a total of $2^{n-2} \cdot n$ x_1, x_n constraints are generated.

We are interested in the interaction of two adjacent sets of breakpoints and thus in the interaction between the constraints corresponding to a_1 and a_n in two adjacent groups. The order of these groups relates to the values of $\frac{p_i}{p_{i-1}}$ and $\frac{a_n}{a_1}$.

Grade: B.

14 September 22, 2014

We have that each time one of the verticies is eliminated new constraints are added between x_1 and each remaining vertex. At the end of the elimination procedure for each k and each subset M' of M we have that there exists a constraint between x_1 and x_n of the form

$$x_1 - a_k \cdot \left(\prod_{M_i \in M'} M_i \right) \cdot x_n \geq c$$

Thus at the end of the variable elimination process we have that there are at most 2^{n-2} groups of n constraints. Now for each pair of adjacent groups we need to compare the intersection of the constraint with a_1 as part of the coefficient in one group and the constraint with a_n as part of the coefficient in the other group. We now need to choose values of $\mathbf{a}$, M, and $\mathbf{c}$ to maximize the number of non-redundant groups.

Grade: C. I don't see the progress. The techniques that you mention are well-known. It is still unclear whether you can get the $\Omega(m^{\log n})$ bound. If not, then you have a polynomial time elimination procedure. There also seems to be little work along the lines of building up the closure algorithm.

15 September 15, 2014

We need a more specific method of causing this grouping of breakpoints. We know that to find the breakpoints we can order the non-redundant constraints in order of slope. Thus if all constraints are of the form $x_i - a_k \cdot x_j \geq c_k$ then we can order the non-redundant x_i, x_j constraints in order of increasing a_k.

One possible way to obtain the desired grouping of breakpoints is to ensure that with every variable elimination, the newly generated x_1, x_n constraints all have larger a_k than the already existing x_1, x_n constraints.

We can then construct a system as follows:

1. For every $i \notin \{1, n\}$ we have n constraints each of the form $x_1 - a_k \cdot x_i \geq c_k$.
2. For every pair $1 < i < j$, we have a single constraint of the form $x_i - M_i \cdot x_j \geq c_{ij}$ for some large $M_i >> \max c_k$.

Thus after each variable elimination the newly introduced x_1, x_n constraints all have their c_k values multiplied by some large M_i. The later these constraints are introduced in the elimination procedure the more M_i values appear. Thus these constraints are grouped as desired. We also have that in each group of constraints the breakpoints appear in the same order. Thus to determine the final number of breakpoints we only need to ensure that a large enough number of these groups remain instead of having to focus on single constraints.

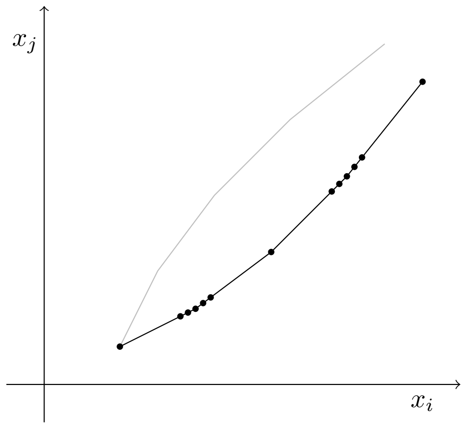
Grade: C+. There is some progress, but now you need to come up with a solution.

16 September 8, 2014

We have a method for precomputation of the closure of a system of generalized difference constraints whose running time depends on the number of breakpoints that variable elimination can produce between each pair of variables x_i and x_j. If this number is polynomial in n and m (the number of starting variables and constraints) then we are done. We also know that the number of x_i, x_j breakpoints depends on the number of non-redundant x_i, x_j constraints that can be derived from the system. We know that an exponential number of x_i, x_j constraints can be derived however many of these constraints may be redundant and thus not contribute to the number of breakpoints.

We will now focus on seeing if there is a set of clauses which can generate a non-polynomial number of non-redundant x_i, x_j constraints.

We may consider a system where we already start with a number of constraints between each pair of variables. These constraints are structured so that the breakpoints are grouped close together and so that eliminating a variable will in some way combine the sets of breakpoints as demonstrated in bellow.



The main issue with trying to determine the number of breakpoints generated by a certain form of system is that all the generated constraints need to be considered. Since the addition of even a single additional constraint can drastically reduce the number of breakpoints in the system.

Grade: C. There does not seem to be much progress since out last meeting.