Some Results concerning G_1 and Polynomial Local Search

Alan Skelley, January 3, 2005.

1 Introduction

We show that the problem of finding witnesses for the restricted quantifiers (those equivalent to existential quantifiers) in a G_1 proof has exactly the same complexity as the local search class polynomial local search (PLS).

2 General Definitions and Lemmas

In this section we introduce some definitions and conventions. First we define local search, and in particular, polynomial local search.

In general a local search problem Π has a set of instances D_Π, which are strings. To each instance $x \in D_\Pi$ there corresponds a set $\mathcal{S}_\Pi(x)$ of solutions, and one standard solution $s_0 \in \mathcal{S}_\Pi(x)$. Each solution $s \in \mathcal{S}_\Pi(x)$ has a cost $f_\Pi(s,x)$ and a neighborhood $\mathcal{N}_\Pi(s,x)$. The search problem is, given an instance, to find a locally optimal solution, which is to say a solution such that none of its neighbors has a better cost. (We may be looking for maximal or minimal cost).

The following is from Yannakakis (1997):

Definition 2.1 (Polynomial Local Search (PLS)) *A local search problem Π is in* PLS *if its instances D_Π and solutions $\mathcal{S}_\Pi(x : x \in D_\Pi)$ are binary strings, there is a polynomial p such that the length of solutions $\mathcal{S}_\Pi(x)$ is bounded by $p(|x|)$, and there are three polynomial-time algorithms A_Π, B_Π, C_Π with the following properties:*

1. *Given a string $x \in \{0,1\}^*$, algorithm A_Π determines whether x is an instance ($x \in D_\Pi$), and in this case it produces some solution $s_0 \in \mathcal{S}_\Pi(x)$.*
2. *Given an instance x and a string s, algorithm B_Π determines whether $s \in \mathcal{S}_\Pi(x)$ and if so, B_Π computes the cost $f_\Pi(s,x)$ of the solution s.*
3. *Given an instance x and a solution s, algorithm C_Π determines whether s is a local optimum, and if it is not, outputs a neighbor $s' \in \mathcal{N}_\Pi(s,x)$ with (strictly) better cost, i.e., $f_\Pi(s',x) < f_\Pi(s,x)$ for a minimization problem, and $f_\Pi(s',x) > f_\Pi(s,x)$ for a maximization problem.*

Now we define a property of quantifiers which will help us state the witnessing problem we wish to solve.

Definition 2.2 (Quantifier Orientations) *The orientation of an* occurrence *of a quantifier is defined as follows:*

- *If ϕ is $\forall x A(x)$ then the orientation of the outermost quantifier occurrence in ϕ is general.*
- *If ϕ is $\exists x A(x)$ then the orientation of the outermost quantifier occurrence in ϕ is restricted.*
- *If ϕ is $\theta \vee \psi$ or $\theta \wedge \psi$ then occurrences of quantifiers in ϕ have the same orientation as they do in the corresponding subformula θ or ψ.*
- *If ϕ is $\neg\psi$ then all occurrences of quantifiers in ϕ have the opposite orientation of their corresponding occurrences in ψ.*
- *Occurrences of quantifiers in succedents of sequents are as described above, and those in antecedents of sequents are the reverse.*

The intuition is that if a sequent were converted into an equivalent formula in prenex form, a restricted quantifier in the original sequent would become an existential one and a general quantifier would become universal.

Definition 2.3 (G_1) *G_1 is the subsystem of the quantified propositional system G where all formulas in all sequents either contain only restricted or only general quantifiers. (i.e. all formulas are Σ_1^q or Π_1^q.)*

3 Witnessing G_1 proofs is no harder than PLS

The witnessing problem we wish to solve is as follows: Given a G_1 proof of a sequent S, and assignments to the free variables $\vec{p}_S$ and generally quantified variables $\vec{x}_S$ (some of which may have the same name) in S (collectively the *input* assignments), find assignments to the restrictedly quantified variables $\vec{y}_S$ in S such that S is satisfied (witnesses or *output* assignments). As formulated this is a total search problem, and we shall now formulate it as a PLS problem G_1W whose instances are the input to the witnessing problem, and whose locally optimal solutions correspond to the witnesses we seek.

Before we do, some motivation is in order. Consider the following (exponential-time) way to find witnesses for the final sequent of a proof: First recursively find appropriate witnesses for the hypotheses, then combine them to obtain a witness for the final sequent. By appropriate witnesses for the hypotheses, we mean that the assignments passed into the witnessing subproblems are chosen so that the resulting witnesses can be easily manipulated to obtain the desired witness. Let us examine how this is done in each case:

Lemma 3.1 *To solve the witnessing problem for a sequent S and an assignment σ in a G_1 proof, it suffices to solve a witnessing problem for each hypothesis of the sequent, and then in polynomial time (in the size of the 2 or 3 sequents involved) we can produce a witness for S.*

Proof: There is one case for each inference rule. In each case we describe the witnessing problem(s) to solve for the hypothesis

- Initial sequents contain no quantifiers so they evaluate to true under any assignment. They are "automatically" witnessed and have no hypotheses.
- If S is obtained by exchange or $\neg$-introduction from S', then the input assignment for S' is obtained by reordering the input assignment to S. The witness for S is obtained by reordering the witness for S'.
- If S is inferred by weakening, $\wedge$: **left** or $\vee$: **right** from S', then solve the witnessing problem for S' with the assignment σ restricted to only those free variables and quantifiers occurring in S'. The resulting witness can be augmented by an assignment of 0 to every new restricted quantifier to obtain a witness for S.
- S is obtained from S' by contraction, deleting a copy of A.
  - If A is quantifier free then a witness for S' with input σ is the witness we seek.
  - If A's quantifiers are restricted, then an input assignment for S is also an input assignment for S'. Solve the witnessing problem, and the result will have witnesses for the quantifiers in each copy of A. This witness contains too many assignments to be a witness for S, since S contains one less copy of A. We choose one of the two sets of assignments for A as follows: Pick the first of the two sets of assignments which falsifies (resp. satisfies) the corresponding copy of A in the antecedent (resp. succedent) of S', and if neither works then pick the first set.
  - If A's quantifiers are general, then an input assignment for S does not contain enough assignments to work for S', since S' contains more general quantifiers. However we may use as input assignment to S' σ modified to include two copies of the assignments to A. The resulting witness for S' is also a witness for S.

- If S is obtained by $\wedge$: **right** or $\vee$: **left**, then we restrict the assignment in the obvious way and obtain witnesses for the hypotheses. We then combine them as follows: We retain the entire witness for the first hypothesis of S. We add the set of assignments to the minor formula in the other hypothesis of S (if any) to obtain a witness for S.

- If S is obtained from S' by the introduction of a general quantifier, then an input assignment to S contains an assignment for that quantifier. S' contains a free variable that S does not. Obtain an input assignment for S' by moving the assignment to the new quantifier so that it becomes an assignment to the old free variable. The resulting witness for S' is a witness for S.

- If S is obtained from S' by the introduction of a restricted quantifier, then S' may contain both quantifiers and free variables which S does not. An input assignment to S is converted into an input assignment to S' by adding an assignment of 0 to every extra free variable and general quantifier. A witness for S is obtained from a witness for S' by evaluating the subformula which was replaced by the newly quantified variable in S and using that value as the assignment to the new variable. Also, all sets of assignments in the witness for S' which are not needed (because those quantifiers do not appear in S) are discarded.

- The final case is when S is obtained using the cut rule with cut formula A. There are 3 subcases:

  - A is quantifier free. In this case, an input assignment for S also works for each hypothesis (possibly augmented by assignments of 0 to free variables in A not occurring in S). Once witnesses for each hypothesis are obtained, clearly one of them also witnesses S (since the formula A must get the same value in each case). Choose the first.
  - A contains restricted quantifiers. If S' is the hypothesis of S in which A occurs on the right then an input assignment to S is also an input assignment to S'. A witness to S' under this input assignment will either satisfy S or A. In the former case no more work is required but in the latter case the satisfying assignment to A is added to the input assignment to S to obtain an input assignment to the other hypothesis, S'', of S, in which A occurs on the left. A witness to S'' under this assignment clearly satisfies S.
  - A contains general quantifiers. Analogous to the above.

■

3.1 The PLS problem G_1W

We may now formally define the PLS problem G_1W as follows:

- Instances: $D_{G_1W} = \{< \pi, \sigma >$: $\pi =< S_1, ..., S_k >$ is a G_1 proof of a sequent $S = S_k$ and σ is an assignment to all the free and generally quantified variables in $S\}$.
- Solutions: $S_{G_1W}(< \pi, \sigma >) = \{< \tau_1, ..., \tau_k >$: k is the number of sequents in π, and each τ_i is a (not necessarily correct) witness to $S_i\}$. (In fact, we haven't even specified under which input assigmnent the witness is intended to work.) These solutions are shorter than π.
- Standard solution: The algorithm A_{G_1W} checks to make sure that its input string is a pair $< \pi, \sigma >$, that π is a correct G_1-proof and that σ is an input assignment to the endsequent of π (i.e. is the correct length). If all the proceeding are true then it outputs the string $< \vec{0}, ..., \vec{0} >$ of all-zero witnessing assignments to every sequent in π.
- Cost function: The algorithm B_{G_1W} on input $< \pi, \sigma >$ and $< \tau_1, ..., \tau_k >$ proceeds as follows:

  1. counter := 0
  2. current := k (k=number of sequents in π)
  3. $\rho := \sigma$
  4. if τ_k witnesses the endsequent of π under assignment σ, then exit with return value 2^k.
  5. for each hypothesis S_j of S_{current}, (in the case of the cut rule, starting from the hypothesis of S_{current} in which the cut formula's quantifiers are general (in the context of the sequent)), do the following:
     (a) compute the correct input assignment ρ' to S_j from ρ (and if necessary from the correct witness to the previous hypothesis, in the case of the cut rule)
     (b) if τ_j witnesses S_j under ρ' then increment counter by 2^j and continue with the next iteration of this loop (i.e. the next hypothesis of S_{current}, if any)
     (c) otherwise, let $\rho := \rho'$, current := j and start the loop again
  6. return the counter

  The intuition behind this algorithm is that $< \tau_1, ..., \tau_k >$ is thought of as the scratch space of the (exponential running time) recursive procedure for computing a witness. The above algorithm simply descends the recursion to the bottom, making note at each level how much progress has been made: how many hypotheses of the corresponding sequent already have correct witnesses computed. If all have correct witnesses then the algorithm terminates. Otherwise, the algorithm knows which hypothesis is the subject of the next-lower recursive call and so moves on to it.

The running total is weighted so that any progress by the recursive algorithm would increase the return value of B_{G_1W}.

- Neighbor of strictly higher cost: The algorithm C_{G_1W} on input $< \pi, \sigma >$ and $< \tau_1, ..., \tau_k >$ proceeds as follows:

  1. current $:= k$ (k=number of sequents in π)
  2. $\rho := \sigma$
  3. if τ_k witnesses the endsequent of π under assignment σ, then the solution is a local optimum; exit.
  4. for each hypothesis S_j of S_{current}, (in the case of the cut rule, starting from the hypothesis of S_{current} in which the cut formula's quantifiers are general (in the context of the sequent)), do the following:
     (a) compute the correct input assignment ρ' to S_j from ρ (and if necessary from the correct witness to the previous hypothesis, in the case of the cut rule)
     (b) if τ_j witnesses S_j under ρ' then continue with the next iteration of this loop (i.e. the next hypothesis of S_{current}, if any)
     (c) otherwise, let $\rho := \rho'$, current $:= j$ and start the loop again
  5. the loop has just exited, meaning that $< \tau_1, ..., \tau_k >$ contains witnesses for each hypothesis of the current sequent, in each case under the correct input assignment as in lemma 3.1, but not however a correct witness for the current sequent itself. Compute τ'_{current} from these and return the new solution $< \tau_1, ..., \tau'_{\text{current}}, ..., \tau_k >$ obtained by substituting the correct τ'_{current} for the incorrect τ_{current} in the input solution

The above algorithm is very similar to the previous one, except that when it locates the bottom of the recursion - the place where the exponential-time recursive algorithm would hypothetically be about to do the next computation - C_{G_1W} performs this computation and returns the new scratch space as the neighbor solution. This new solution clearly has higher cost as evaluated by B_{G_1W}.