On Approximation Preserving Reductions: Complete Problems and Robust Measures (Revised Version)

Pekka Orponen* Heikki Mannila
Department of Computer Science, University of Helsinki
Teollisuuskatu 23, SF–00510 Helsinki, Finland

2 May 1990

Abstract

We investigate the well-known anomalous differences in the approximability properties of NP-complete optimization problems. We define a notion of polynomial time reduction between optimization problems, and introduce conditions guaranteeing that such reductions preserve various types of approximate solutions. We then prove that a weighted version of the satisfiability problem, the traveling salesperson problem, and the zero-one integer programming problem are in a strong sense approximation complete for the class of NP minimization problems. Finally, we discuss the reasons that cause the standard relative error approximation quality measure to break down in computationally simple problem transformations, and give a general construction for producing quality measures that are more robust with respect to an arbitrary given class of invertible transformations.

1 Introduction

Since many optimization problems are *NP*-complete, and thus probably not solvable in polynomial time, approximate solution methods for them are of great interest. An *approximation algorithm* (cf. [9]) for an optimization problem is a polynomial time algorithm that always produces feasible, but not necessarily optimal solutions. The quality of the approximate solutions is commonly measured by their relative error; if s is an approximate solution, s^* is a corresponding optimal solution, and c is the cost function, the quality of s is taken to be

$$\mu_r(s) = \frac{c(s) - c^*(s)}{c(s^*)}$$

(for definiteness, we assume we are dealing with a minimization problem). An approximation algorithm is said to have *bounded error*, if there is an $\epsilon > 0$ such that for all solutions s produced by the algorithm, $\mu_r(s) \leq \epsilon$. A *polynomial time approximation scheme* (PTAS) is a sequence of algorithms such that for every $\epsilon > 0$, some algorithm in the sequence guarantees $\mu_r(s) \leq \epsilon$ for all s.

*Work supported by the Academy of Finland.

While many practical and efficient approximation algorithms have been devised (cf. [9, 16]), a number of theoretical issues remain unsettled. Perhaps the most perplexing one concerns the observed differences in the approximability properties of *NP*-complete problems: some problems have a PTAS (e.g. the knapsack problem); some have bounded error approximations but a PTAS only if $P = NP$ (e.g. the bin packing problem[1]); and others do not have even bounded error approximations unless $P = NP$ (e.g. the traveling salesperson problem, when not constrained by the triangle inequality) [9, 19]. Such differences appear rather unexpected, as the set recognition versions of these problems are so strongly interreducible (in fact, polynomial time isomorphic [4]).

Several authors have suggested that these differences be investigated by considering reductions between problems that are more sensitive to optimization structure than the standard polynomial time many-one ($\leq_m^p$) reductions between sets [3, 11, 13, 17]. (For a survey of this work, see [6].) A natural optimization analogue of $\leq_m^p$-reducibility (actually, $\leq_{1-tt}^p$-reducibility) was defined by Ko in his thesis [11], but this work has unfortunately remained little known. Most of the other definitions suggested in the literature have failed to capture some aspect of the situation: the reductions in [13] only require that optimal, not approximate, solutions are preserved, whereas the reductions in [3] are overly strict about preserving the complete structure of the solution space. The "ratio preserving" reductions of [17] make the stringent requirement that the optimal cost associated with any problem instance be preserved up to a multiplicative constant, but in the "constructive" version of the reductions attention is again only paid to the optimal, not approximate solutions.

In Section 2 of this paper we give our definitions of optimization reductions that preserve different types of approximations. These are essentially the same as Ko's, although independently conceived. In addition to our work, these notions have been used by Crescenzi and Panconesi in related studies [8], and very similar definitions have been put forth recently by Berman and Schnitger in [5].

The main results of our paper are contained in Section 3, where we prove that a weighted version of the satisfiability problem, the traveling salesperson problem, and the zero-one integer programming problem are in a strong sense approximation-preserving complete for the class of *NP* minimization problems. Results analogous to ours on the weighted satisfiability problem have appeared in the literature before [3, 17], but the results concerning the approximation completeness of the TSP and zero-one programming are the first truly natural ones of this kind. (The completeness of zero-one programming has been noted, independently, also by Berman and Schnitger [5].) The significance of such completeness results is that they provide an *explanation* for the unapproximability of certain problems. For instance, whereas the standard unapproximability proof for the TSP is based on an uninformative weight-assignment trick we now know that the "reason" for the unapproximability of this problem lies in its combinatorial richness: the problem of approximating any *NP* minimization problem can be encoded into the problem of approximating the TSP. A notion of completeness is of course also a potentially very useful tool: as an example, the fact that zero-one integer programming is unapproximable unless $P = NP$, though hardly surprising, seems not to have been noted in the literature before.

In Section 4 we investigate further into what causes the standard relative error quality measure to break down even in very simple reductions. Intuitively, we locate the reason in

[1]To be precise, bin packing *does* have a PTAS in the asymptotic sense that for every ϵ, some algorithm guarantees $\mu_r(s) \leq \epsilon$ for all *sufficiently large* s [10].

the incongruence between the computational nature of the reductions and the arithmetical nature of the measure. The question then arises whether it is in general possible, given some class of simple reductions, to have an approximation quality measure that respects these reductions, in the sense of being invariant under them. We give a general construction for producing such a measure, when all the solution mappings associated with the reductions are *invertible*. This part of our study was inspired by similar work done in the context of zero-one programming problems by Zemel [20]. In some sense, our results are a generalization of his, but the techniques used are completely different.

2 Optimization problems, approximations, and reductions

Following [9], we represent an *optimization problem* Π as a triple (D, S, c), where D is the set of *instances*, S is the set of *feasible solutions*, and c is the solution *cost* function. To each instance $x \in D$ there is associated a finite subset $S(x)$ of the solutions, and the cost function maps pairs $\langle x, s\rangle$, where $x \in D$ and $s \in S(x)$, to nonnegative integers. Given an instance x, the objective in solving the problem is to locate a solution $s^* \in S(x)$ with minimal cost. (For simplicity, we consider here only minimization problems.) Thus, the *optimal cost* associated with an instance $x \in D$ is

$$c^*(x) = \min\{c(x, s) : s \in S(x)\},$$

and the set of *optimal solutions* is

$$Opt(x) = \{s^* \in S(x) : c(x, s^*) = c^*(x)\}.$$

For uniformity, we assume that all problems are coded over the binary alphabet $\Sigma = \{0, 1\}$, so that D and S are subsets of Σ^*. Further, we assume that for each $s \in S$ there is only one $x \in D$ such that $s \in S(x)$; we denote this x by $I(s)$, and assume that the mapping I can be computed in polynomial time. (This partitioning of the solution space may of course be effected artificially by replacing the set S by the set $S' = \{\langle x, s\rangle : x \in D, s \in S(x)\}$, where $\langle , \rangle$ is some standard pairing function.) This assumption enables us to use the abbreviation $c(s)$ for the cost $c(I(s), s)$, and to say that a solution s is *optimal* when $s \in \mathrm{Opt}(I(s))$.

Problem $\Pi = (D, S, c)$ is an *NP optimization problem*, if the sets D and S are in P, the function c is polynomial time computable, and there is a polynomial p such that for all s and x, the condition $s \in S(x)$ implies that $|s| \leq p(|x|)$. (Here $|w|$ denotes the length of a string $w \in \Sigma^*$.) These conditions imply that the predicate "$s \in S(x)$" is computable in polynomial time, and the set

$$L_\Pi = \{\langle x, k\rangle : x \in D, c^*(x) \leq k\}$$

is in *NP*. (Integers are represented in binary notation, without leading zeros.) An *NP* optimization problem Π is said to be *NP-complete*, if the associated set L_Π is *NP*-complete.

An additional assumption concerning *NP* optimization problems we make is that for each $x \in D$, some element $\mathrm{triv}(x) \in S(x)$ can be computed in polynomial time. While some natural problems do not have this property, it can usually be ensured by admitting, perhaps artificially, some trivial feasible solutions.

An *approximation algorithm* for a problem Π is any polynomial time algorithm $A : D \to S$, such that for each $x \in D$ we have $A(x) \in S(x)$. Several measures have been introduced for assessing the quality of solutions produced by such algorithms. The most commonly used is

the *relative error* measure, defined as

$$\mu_r(s) = \frac{c(s) - c^*(I(s))}{c^*(I(s))},$$

if $c(I(s)) > 0$, and as $\mu_r(s) = c(s)$ if $c(I(s)) = 0$. Other examples are the *absolute error* measure

$$\mu_a(s) = c(s) - c^*(I(s)),$$

and the *normalized relative error* measure [1, 3]

$$\mu_n(s) = \frac{c(s) - c^*(I(s))}{c^0(I(s)) - c^*(I(s))},$$

where $c^0(x) = \max\{c(s) : s \in S(x)\}$.

Still other quality measures, some of them of a very different type, have been considered by Zemel [20] in the context of zero-one integer programming problems.

In general, we may define a *measure of approximation quality* (for a problem Π) to be a function $\mu : S \to \mathcal{R}^+$ such that for optimal s^* we have $\mu(s^*) = 0$. The measure is *unbounded* if it may assume arbitrarily large values, and *cost-respecting* if $c(s_1) \leq c(s_2)$ implies that $\mu(s_1) \leq \mu(s_2)$. All the measures mentioned above are cost-respecting, and all except μ_n are unbounded (for appropriate Π).

Given a function $f : N \to \mathcal{R}^+$, an approximation algorithm A for a problem Π is said to be an $f(n)$-*approximation algorithm* (with respect to a quality measure μ), if for all $x \in D$, $\mu(A(x)) \leq f(|x|)$. We say that a problem has a *bounded approximation* if it has an ϵ-approximation algorithm for some constant $\epsilon > 0$. A stronger requirement is that a problem have a *polynomial time approximation scheme* (PTAS). This is a sequence $A_1, A_2, \ldots$ of approximation algorithms such that for every constant $\epsilon > 0$, some algorithm A_i in the sequence provides an ϵ-approximation.

Let $\Pi_1 = (D_1, S_1, c_1)$ and $\Pi_2 = (D_2, S_2, c_2)$ be two optimization problems. A pair (f, g) of polynomial time computable functions is an *optimization reduction* from Π_1 to Π_2 if f maps D_1 to D_2 and g maps $D_1 \times S_2$ to S_1, so that for each $x \in D_1$ and $t \in S_2(f(x))$ we have $g(x, t) \in S_1(x)$. In addition, we require that for each $x \in D_1$ and $t^* \in \mathrm{Opt}(f(x))$ we have $g(x, t^*) \in \mathrm{Opt}(x)$. If there is an optimization reduction from Π_1 to Π_2, we denote $\Pi_1 \leq_o^p \Pi_2$. Clearly, if $\Pi_1 \leq_o^p \Pi_2$ and Π_2 can be solved in polynomial time, then so can Π_1. The $\leq_o^p$-reducibility is the analogue for optimization problems of the polynomial time many-one reducibility (more precisely, polynomial time one-query truth table reducibility) between sets.

We now consider conditions on optimization reductions to guarantee that they preserve various types of approximations. Let (f, g) be a reduction from problem $\Pi_1 = (D_1, S_1, c_1)$ to problem $\Pi_2 = (D_2, S_2, c_2)$, and let μ be an approximation quality measure. (Actually we should have a different measure for each problem, but context will always make clear which one is meant.) The strongest requirement is that the reduction be *strict*, by which we mean that $\mu(g(x, t)) \leq \mu(t)$ holds for all $x \in D_1$ and $t \in S_2(f(x))$. The reduction is *bounded* if for every $r > 0$ there exists an $R > 0$ such that $\mu(t) \leq r$ implies $\mu(g(x, t)) \leq R$, for all $x \in D_1$, $t \in S_2(f(x))$. The reduction is *continuous* if for every $R > 0$ there exists an $r > 0$ such that for almost all $x \in D_1$ and $t \in S_2(f(x))$, $\mu(t) \leq r$ implies $\mu(g(x, t)) \leq R$.

Of course, every strict reduction is also bounded and continuous. The motivation for the weaker notions is the following simple result, showing that bounded and continuous reductions preserve bounded approximations and approximation schemes, respectively.

Proposition 2.1 *(i) If* $\Pi_1 \leq_o^p \Pi_2$ *via a bounded reduction, and* Π_2 *has a bounded approximation, then so does* Π_1.

(ii) If $\Pi_1 \leq_o^p \Pi_2$ *via a continuous reduction, and* Π_2 *has a PTAS, then so does* Π_1. □

To build a reasonable theory, we need to verify that our reductions are transitive.

Proposition 2.2 *If* $\Pi_1 \leq_o^p \Pi_2$ *via a strict (bounded, continuous) reduction, and* $\Pi_2 \leq_o^p \Pi_3$ *via a strict (bounded, continuous) reduction, then also* $\Pi_1 \leq_o^p \Pi_3$ *via a strict (bounded, continuous) reduction.* □

3 Approximation complete problems

Let C be a class of optimization problems and μ an approximation quality measure. A problem Π is *(strictly, bounded, continuous) complete* for the class C w.r.t. measure μ if $\Pi \in C$ and every $\Pi' \in C$ is reducible to Π via a (strict, bounded, continuous w.r.t. μ) optimization reduction. Clearly strictly complete problems are also bounded and continuous complete.

Since there exist *NP* optimization problems that do not have bounded approximations unless $P = NP$, no bounded complete *NP* optimization problem can have a bounded approximation unless $P = NP$. A similar situation holds w.r.t. PTAS for problems that are continuous complete in the class *APX*, the class of *NP* optimization problems with bounded error approximations. The class *APX* and its complete problems have been studied by Crescenzi and Panconesi in [8], and a very interesting subclass *MAX NP* of *APX*, together with an appropriate completeness notion, was introduced by Papadimitriou and Yannakakis in [18] (see also [15]). We concentrate here on the class of all *NP* minimization problems.

Theorem 3.1 *The following problem is strictly complete for the class of NP minimization problems w.r.t. any cost-respecting quality measure* μ.

Weighted Satisfiability (WSAT)

Instance: *Boolean formula* F*, with nonnegative integer weights* $w(p)$ *on the variables appearing in* F.

Objective: *Find a truth assignment* $t(p)$ *to the variables that satisfies* F *and minimizes*

$$w(t) = \sum_{t(p)=\text{true}} w(p).$$

We consider also the assignment $\text{triv}(F, w)$ *that sets all the variables true to be feasible, even though it possibly does not satisfy* F.

Proof. It is clear that WSAT is an *NP* minimization problem, so we have to show that any other such problem strictly reduces to WSAT. The construction here is a modification of Cook's proof [7, 9], similar to the ones appearing in [2, 17][2]. Let $\Pi = (D, S, c)$ be an *NP* minimization problem, and let p be a polynomial such that for each $x \in D$ the condition $s \in S(x)$ implies that $|s| \leq p(|x|)$. Let M be an *NP* machine that on a given instance generates all its feasible solutions and their costs, operating as follows.

[2]The priority for claiming this type of result apparently belongs to the preliminary version of [17], although proofs did not appear until a few years later, concurrently in [17] and [2].

On input x:
if $x \notin D$ then reject;
generate any s, $|s| \leq p(|x|)$;
if $s \notin S(x)$ then reject;
print s;
print $c(s)$;
accept.

Consider the standard Cook reduction from the set accepted by M to the set of satisfiable Boolean formulas. Denote by F_x the formula that describes the accepting computations of machine M on input x. Let $c_n, \ldots, c_0$ be the variables in F_x that correspond to the tape squares on which M prints the value $c(s)$ (in binary), so that in a satisfying assignment, c_i is true if and only if bit i from the right of $c(s)$ is 1. A strict reduction (f, g) from Π to WSAT may now be obtained by defining, for $x \in D$,

$$f(x) = \langle F_x, w \rangle, \quad \text{where} \begin{cases} w(c_i) = 2^i, & \text{for } i = 0, \ldots, n, \\ w(p) = 0, & \text{for other variables } p \text{ in } F_x; \end{cases}$$

and for $x \in D$ and $t \in S_{\text{WSAT}}(f(x))$,

$$g(x,t) = \begin{cases} \text{triv}(x), \text{ if } t = \text{triv}(f(x)), \\ \text{the string } s \text{ printed by } M \text{ according to } t, \text{ otherwise.} \end{cases}$$

It is straightforward to check that f maps D_Π to D_{WSAT} and g maps $D_\Pi \times S_{\text{WSAT}}$ to S_Π so that for each $x \in D_\Pi$ and $t \in S_{\text{WSAT}}(f(x))$, $g(x,t) \in S_\Pi(x)$. Thus it remains to verify that the reduction is strict.

There are two cases: if $t = \text{triv}(f(x))$, then

$$c(g(x,t)) = c(\text{triv}(x)) \leq 2^{n+1} - 1 = w(t),$$

and because μ is cost-respecting, $\mu(g(x,t)) \leq \mu(t)$.

Otherwise, the truth assignment t satisfies F_x, and so represents an accepting computation of M on x. Thus $s = g(x,t)$ is a feasible solution for x, and

$$c(s) = \sum_{t(c_i)=\text{true}} 2^i = w(t).$$

Hence also $\mu(s) = \mu(t)$. □

Corollary 3.2 *The problem* W3SAT, *i.e.* WSAT *restricted to 3-cnf formulas [9, p. 48], is strictly complete for the class of NP minimization problems w.r.t. any cost-respecting measure* μ.

Proof. The standard proof for the *NP*-completeness of the unweighted 3SAT problem [9, pp. 48–49] shows how to transform the formula F_x above into a 3-cnf formula that is satisfiable if and only if F_x is. The auxiliary variables introduced in the process may be given zero weight, and so they have no effect on the approximation structure. □

Theorem 3.3 *The traveling salesperson problem, defined below, is strictly complete for the class of NP minimization problems w.r.t. any cost-respecting quality measure* μ.

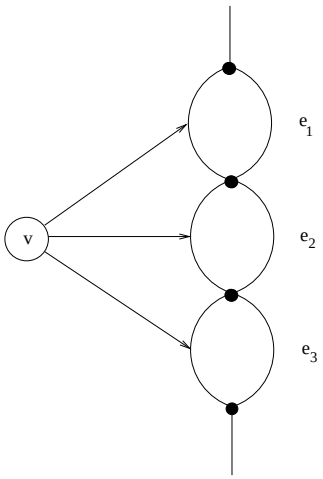

Figure 1: Graph gadget representing a clause $e_1 \vee e_2 \vee e_3$.

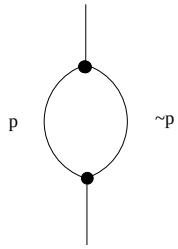

Figure 2: Graph gadget representing a variable p.

Traveling Salesperson (TSP)

Instance: *Complete undirected graph G, with nonnegative integer weights ("distances") $d(e)$ on the edges.*

Objective: *Find a tour, i.e. a Hamiltonian circuit C in G that minimizes*

$$d(C) = \sum_{e \in C} d(e).$$

Proof. We modify the construction given in [16, pp. 366–370] for reducing 3SAT to the Hamiltonian Circuit problem. The reduction is based on a method for transforming a 3-cnf Boolean formula F to an undirected graph G, so that G has a Hamiltonian circuit if and only if F is satisfiable. The graph is built up by combining graph "gadgets" that correspond to the parts of F. For each of the clauses in F there is a gadget represented as shown in Figure 1. The inner workings of this component are not important; its essential property is that in a Hamiltonian circuit through a graph, at least one of the edges e_1, e_2, e_3 must remain untraversed (this corresponds to the true literal in a satisfied clause). The gadget corresponding to a variable p is simple; it is shown in Figure 2. The idea is that in a Hamiltonian circuit, exactly one of the edges denoted p and $\neg p$ will be taken. The clause and variable components are combined by means of a gadget denoted as shown in Figure 3.

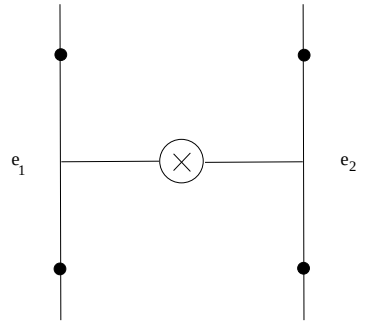

Figure 3: Graph gadget forcing the traversal of either e_1 or e_2.

This has the property that in a Hamiltonian circuit, exactly one of the edges e_1 and e_2 is traversed.

An example should make clear how the transformation works. Let F be the formula

$$(p_1 \vee \neg p_2 \vee \neg p_3) \wedge (\neg p_1 \vee p_2 \vee p_3);$$

the corresponding graph G is shown in Figure 4. We have there outlined a Hamiltonian circuit that corresponds to the satisfying assignment $t(p_1) = t(p_2) = \text{true}$, $t(p_3) = \text{false}$.

A strict reduction (f, g) from W3SAT to TSP may now be obtained by defining

$$f(F, w) = \langle G, d \rangle,$$

where G is the graph obtained from formula F as above (say with edge set V), plus a number of added edges (say V^+) to make the graph complete. The weights on the edges of G are set as follows:

$$d(e) = \begin{cases} w(p), & \text{if } e \text{ corresponds to a variable } p \text{ in } F; \\ 0, & \text{if } e \text{ is some other edge in } V; \\ w(\text{triv}(F, w)), & \text{if } e \text{ is an edge in } V^+. \end{cases}$$

The function g is defined as

$$g(\langle F, w \rangle, C) = \begin{cases} \text{the truth assignment } t \text{ determined by how } C \\ \qquad \text{traverses the variable components in } G, \text{ if } C \\ \qquad \text{traverses only edges in } V; \\ \text{triv}(F, w), \text{ otherwise.} \end{cases}$$

It can be seen that if we have a tour C in the complete graph that traverses only edges in V, then the truth assignment $g(\langle F, w \rangle, C)$ satisfies F and we have $d(C) = w(g(\langle F, w \rangle, C))$. On the other hand, if the tour C traverses some edge in V^+, then $g(\langle F, w \rangle, C) = \text{triv}(F, w)$, and $d(C) \geq w(\text{triv}(F, w)) = w(g(\langle F, w \rangle, C))$. Since the measure μ is cost-respecting, this shows that the reduction (f, g) is strict.

Actually, we have glossed over one small complication in the proof. The arcs in the gadget of Figure 3 consist really of *series* of edges in the graph; in detail the component looks as in Figure 5. This causes a problem if such a composite arc corresponds to a variable p with nonzero weight. But in this case we can place the weight $w(p)$ on either one of the edges starting from the ends of the arc (e or e' in Figure 5). □

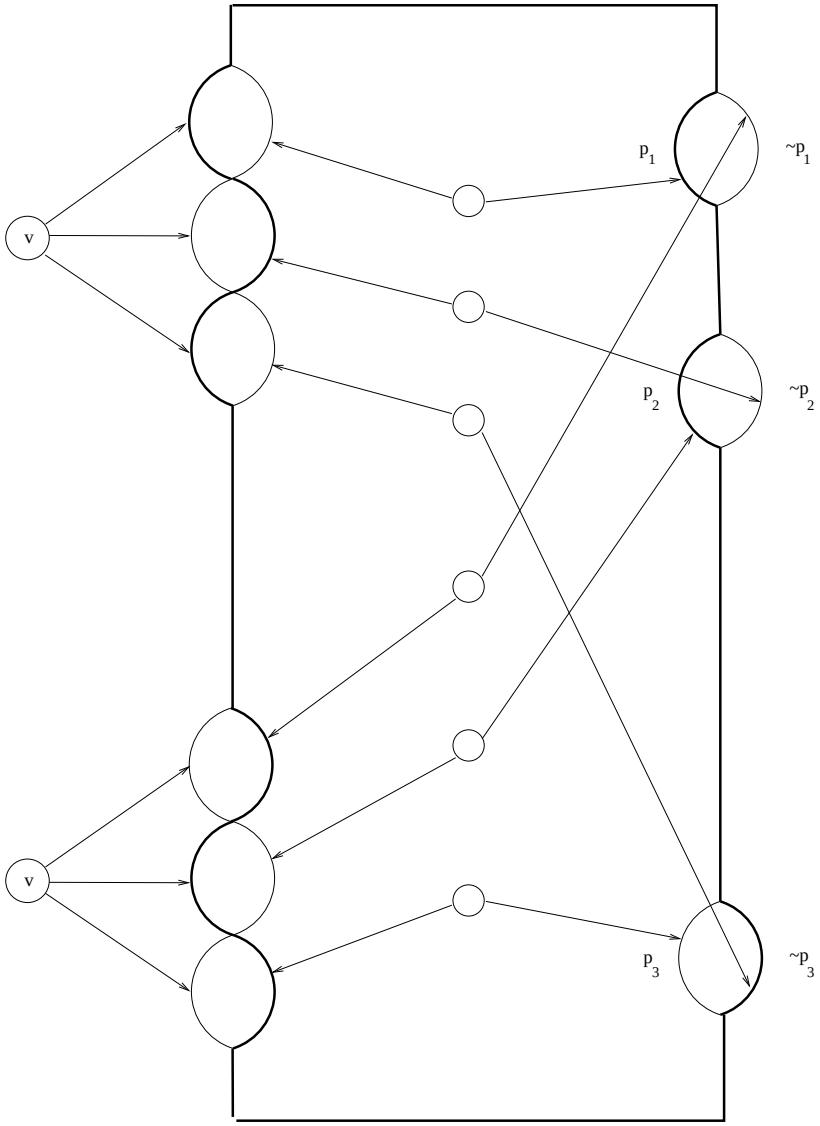

Figure 4: Graph representation of a satisfiable formula.

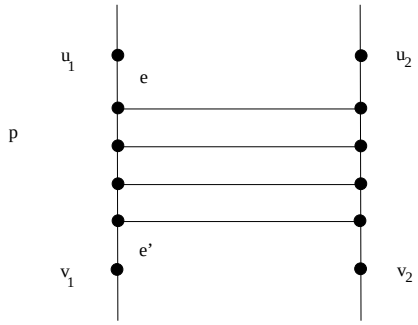

Figure 5: Detailed implementation of the edge choice gadget.

Theorem 3.4 *The zero-one integer programming problem, defined below, is strictly complete for the class of NP minimization problems w.r.t. any cost-respecting quality measure μ.*

Zero-One Programming (ZOP)

Instance: *$m \times n$ integer matrix A, integer m-vector b, positive integer n-vector c.*

Objective: *Find a zero-one n-vector x that satisfies the requirements $Ax \geq b$ and minimizes the product $c^T x$.*

Here again it is in general difficult to find feasible solutions, so we just set artificially $\mathrm{triv}(A, b, c) = (1, 1, \ldots, 1)^T$.

Proof. Reducing W3SAT to ZOP is very easy; an example should suffice to explain the transformation (the details may be found in [16, pp. 314–315]). Given a cnf Boolean formula F with weights w, the corresponding ZOP instance will have a variable for each variable in F, and a row in the requirements matrix A for each clause in F. The cost vector c is obtained directly from the weights w. Variable value 1 represents "true" and 0 represents "false". For example, assume that F is the formula

$$(p_1 \vee \neg p_2 \vee \neg p_3) \wedge (\neg p_1 \vee p_2 \vee p_3),$$

with $w(p_1) = 2$, $w(p_2) = 3$, $w(p_3) = 0$. Then the corresponding ZOP instance is

$$\begin{array}{l} \text{minimize } 2p_1 + 3p_2 + 0p_3 \\ \text{subject to:} \\ \qquad \begin{array}{lcl} p_1 + (1 - p_2) + (1 - p_3) & \geq & 1 \\ (1 - p_1) + p_2 + p_3 & \geq & 1 \\ p_1, p_2, p_3 & = & 0, 1. \end{array} \end{array}$$

Or, in matrix form:

$$A = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \end{pmatrix} \qquad b = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \qquad c = \begin{pmatrix} 2 \\ 3 \\ 0 \end{pmatrix}.$$

This transformation can easily be formulated as a strict reduction. □

4 Robust quality measures

The preceding formulations provide some insight into why even the simplest reductions between optimization problems may sometimes behave so badly with respect to approximations. The intuitive reason is that a *computationally* simple reduction need not be simple in the *arithmetic* sense of being bounded w.r.t. the relative error measure μ_r. Such incongruence may indicate that μ_r is not an appropriate quality measure for the context. There may be some operations which intuitively do not change a problem instance very much, yet do have a large effect on μ_r; or the whole notion of numerical cost may be inappropriate, and the quality of a solution would better be measured by how far it is from optimal in some structural or computational sense.

Consider, e.g., the context of trying to solve the satisfiability problem by a backtrack method. Part of the problem may there be viewed as that of trying to maximize the number of variables that can be assigned a truth value without making the given formula false. In this case it would be quite natural to consider as part of the quality of a partial truth assignment (i.e., an approximate solution), whether or not it can be extended to a complete satisfying assignment.

Similar objections to the μ_r measure have been considered before by Zemel, who in [20] studied zero-one programming problems and a class of problem transformations natural in that context. He proved that measures whose values are not much affected by these transformations indeed exist, though they may not be very natural.

In this section we follow Zemel's ideas in a general setting. Let us say that a quality measure μ is $\mathcal{R}$-*robust*, for a class of optimization reductions $\mathcal{R}$, if all the reductions in $\mathcal{R}$ are bounded with respect to μ. Our question is then: given a class $\mathcal{R}$ of optimization reductions, can we always find a quality measure μ that is $\mathcal{R}$-robust?

In addition to robustness, though, one would like to require that the measure μ be in some sense "interesting". For instance, if all the reductions $(f, g) \in \mathcal{R}$ are length-increasing (i.e., $|f(x)| \geq |x|$ for all x), then the following trivial measure is unbounded and $\mathcal{R}$-robust:

$$\mu(s) = \begin{cases} 0, \text{ if } s \text{ is optimal,} \\ |I(s)| + |s|, \text{ otherwise.} \end{cases}$$

Obviously, this measure does not give us any information about the relative quality of the suboptimal solutions.

At present, we cannot suggest any criterion for distinguishing between interesting and uninteresting measures. What we can do is give a fairly natural construction of a nontrivial $\mathcal{R}$-robust measure in the special case that all the reductions in $\mathcal{R}$ are *invertible*. By this we mean that for any reduction $(f, g) \in \mathcal{R}$, there is an efficiently computable function h such that for all $x \in D_1$ and $t \in S_2(f(x))$ we have $h(g(x, t)) = t$. Let us call such an h a *left inverse for g restricted by f*. Admittedly, the requirement of invertibility is quite restrictive: to rephrase, we are requiring that for any $x \in D_1$, g maps the feasible solutions of $f(x)$ one-to-one invertibly into the feasible solutions of x. Clearly, only the most similar of problems may be expected to be related by such reductions; on the other hand, we should expect interesting robust measures to exist precisely when the problems considered *are* very similar. Also Zemel considers only invertible reductions in [20].

Let $\mathcal{F} = \{f_i\}$ be a recursively presented (r.p.) class of recursive functions with universal function F (i.e. F is a computable function such that $F(i, x) = f_i(x)$ for all i and x). We say that $\mathcal{F}$ is *sufficiently rich* if it contains the constant and pairing functions, all the I-functions for the problems we are considering, and is closed under composition.

Let $\mathcal{F}$ be a sufficiently rich r.p. class of functions on Σ^*, with universal function F. For strings x, y, we define the F-*complexity of x relative to y* as (cf. [12, 14]):

$$K^F(x|y) = \min\{|i| : F(i, y) = x\}.$$

This value is always defined, since $\mathcal{F}$ contains all the constant functions.

Given an optimization problem, we can define the following "information distance" quality measure for its solutions:

$$\mu^F_{\text{inf}}(s) = \begin{cases} 0, \text{ if } s \text{ is optimal,} \\ \min\{K^F(s^*|s) : s^* \in \text{Opt}(I(s))\}, \text{ otherwise.} \end{cases}$$

The idea here is that of measuring how far, computationally, a solution s is from optimal — more precisely, how many extra bits of information does F need, given s, to compute an optimal solution s^*. The measure is recursively computable if the optimality predicate is, and unbounded if and only if no function in $\mathcal{F}$ can directly compute optimal solutions from feasible ones[3]. Intuitively, it is to be expected that the quality of a solution with respect to this measure cannot be much affected by the functions in $\mathcal{F}$.

Theorem 4.1 *Let $\mathcal{R}$ be a class of optimization reductions, and let $\mathcal{F}$ be a sufficiently rich r.p. class of functions with universal function F. Assume that for any $(f,g) \in \mathcal{R}$, $\mathcal{F}$ contains g, and also a left inverse for g restricted by f. Then the measure $\mu^F_{\inf}$ is $\mathcal{R}$-robust.*

Proof. Let $(f,g) : \Pi_1 \leq^p_o \Pi_2$ be a reduction in $\mathcal{R}$. We shall show that there is a function c such that for any $x \in D_1$ and $t \in S_2(f(x))$,

$$\mu^F_{\inf}(g(x,t)) \leq c(\mu^F_{\inf}(t));$$

the boundedness of the reductions follows directly from this.

Let i be an index (w.r.t. F) for the I-function of Π_1 (i.e., $f_i(s) = I(s)$ for all $s \in D_1$). Let $h \in \mathcal{F}$ be a left inverse for g restricted by f. For any index p, let p' be the smallest index for the function

$$f_{p'}(s) = g(f_i(s), f_p(h(s)));$$

by the closure properties of $\mathcal{F}$, such a p' always exists. Define

$$c(n) = \begin{cases} 0, \text{ if } n = 0, \\ \max\{|p'| : |p| = n\}, \text{ otherwise.} \end{cases}$$

Consider then some $x \in D_1$, $t \in S_2(f(x))$. If t is optimal, then so is $g(x,t)$, and $\mu^F_{\inf}(g(x,t)) = \mu^F_{\inf}(t) = 0 = c(\mu^F_{\inf}(t))$. Thus, let us assume that t is not optimal. Let p be the smallest index such that $F(p,t) = t^*$ for some $t^* \in Opt(f(x))$ (hence $\mu^F_{\inf}(t) = |p|$). Then

$$f_{p'}(g(x,t)) = g(x, f_p(h(g(x,t)))) = g(x,t^*) \in Opt(x),$$

and so

$$\mu^F_{\inf}(g(x,t)) \leq |p'| \leq c(|p|) = c(\mu^F_{\inf}(t)). \ \square$$

Corollary 4.2 *Let $F(i,x)$ be any universal function for the class P (e.g., the one given by the standard presentation of P via "clocked" Turing machines.) Then the measure $\mu^F_{\inf}$ is robust with respect to the class of polynomial time computable, polynomial time invertible optimization reductions.* $\square$

[3]Actually, to prove the "if" direction here, we need to assume also that $\mathcal{F}$ contains for infinitely many $k \geq 0$ a function f^*_k such that for all s,

$$c(f^*_k(s)) = \min\{c(f_0(s)), \ldots, c(f_k(s))\}.$$

Any natural enumeration (in terms of Turing machines, e.g.) will have this "finite minimum" property.

5 Concluding remarks

We have studied reductions between optimization problems that preserve the quality of approximations. We showed that a weighted version of the satisfiability problem, the traveling salesperson problem, and the zero-one integer programming problem are complete in the class of *NP* minimization problems with respect to such reductions. An important point in the proofs was the ability to code the cost functions of arbitrary *NP* minimization problems into the cost functions of these problems. Thus, an intriguing open question concerns the completeness of apparently hard to approximate problems whose instances do not explicitly contain numbers, such as graph coloring and the independent set problem. (Some progress on these issues has recently been reported in [5, 15].) It is known, theoretically, that if $P \neq NP$, then incomplete *NP* optimization problems do exist [8].

We have also studied the existence of measures appropriate for measuring the computational or structural quality of approximate solutions, the basic idea being that problem transformations simple in some framework should not change the values of a quality measure appropriate for that framework. Using the notion of program-size complexity, we have shown how to obtain a quality measure whose values are fairly invariant with respect to any given recursively presentable class of invertible problem transformations. It remains an open question whether interesting invariant quality measures can be obtained without the invertibility assumption. Another open area concerns the existence of natural computational or structural measures: can we find examples of easy-to-compute problem-specific measures that would be invariant with respect to some small, well-understood class of reductions (cf. [20])?

References

[1] A.Aiello, E.Burattini, A.Massaroti, F.Ventriglia, A new evaluation function for approximation algorithms, *Sem. IRIA* (1977).

[2] G. Ausiello, A. D'Atri, M. Protasi, Lattice-theoretical ordering properties for *NP*-complete optimization problems, *Fundamenta Informaticae 4* (1981) 83–94.

[3] G. Ausiello, A. D'Atri, M. Protasi, Structure preserving reductions among convex optimization problems, *J. Comput. System Sciences 21* (1980) 136–153.

[4] L. Berman, J. Hartmanis, On isomorphism and density of *NP* and other complete sets, *SIAM J. Comput. 6* (1977) 305–322.

[5] P. Berman, G. Schnitger, On the complexity of approximating the independent set problem, in: *Proc. 6th Ann. Symp. on Theoretical Aspects of Computer Science*, LNCS 349 (Springer-Verlag, Berlin, 1989) 256–268.

[6] D. Bruschi, D. Joseph, P. Young, A Structural Overview of *NP* Optimization Problems, Technical report #861, Computer Sciences Department, University of Wisconsin – Madison (July 1989).

[7] S. A. Cook, The complexity of theorem-proving procedures, in: *Proc. 3rd Ann. ACM Symp. on Theory of Computing* (ACM, New York, 1971) 151–158.

[8] P. Crescenzi, A. Panconesi, Completeness in approximation classes, in: *Proc. Intern. Conf. on Fundamentals of Computation Theory*, LNCS 380 (Springer-Verlag, Berlin, 1989) 116–126.

[9] M. R. Garey, D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (W. H. Freeman, San Francisco, 1979).

[10] N. Karmarkar, R. Karp, An efficient approximation scheme for the one-dimensional bin-packing problem, in: *Proc. 23rd Ann. IEEE Symp. on Foundations of Computer Science* (IEEE, New York, 1982) 312–320.

[11] K. Ko, Computational Complexity of Real Functions and Polynomial Time Approximations, Ph.D. Thesis, Ohio State Univ., 1979.

[12] A. N. Kolmogorov, Three approaches to the quantitative definition of information, *Probl. Inform. Transm. 1* (1965) 1–7.

[13] M. Krentel, The complexity of optimization problems, *J. Comput. System Sciences 36* (1988) 490–509.

[14] M. Li, P. Vitányi, Two decades of applied Kolmogorov complexity, in: *Proc. 3rd Ann. Symp. on Structure in Complexity Theory* (1988) 80–101.

[15] A. Panconesi, D. Ranjan, Quantifiers and approximation, in: *Proc. 22nd Ann. ACM Symp. on Theory of Computing* (to appear).

[16] C. H. Papadimitriou, K. Steiglitz, *Combinatorial Optimization, Algorithms and Complexity* (Prentice-Hall, New York, 1981).

[17] A. Paz, S. Moran, Non-deterministic polynomial optimization problems and their approximations, *Theoret. Comput. Sci. 15* (1981) 251–277. Preliminary version in: *Proc. 4th Intern. Colloq. on Automata, Languages, and Programming*, LNCS 52 (Springer-Verlag, Berlin, 1977) 370–379.

[18] C. H. Papadimitriou, M. Yannakakis, Optimization, approximation, and complexity classes, in: *Proc. 20th Ann. ACM Symp. on Theory of Computing* (ACM, New York, 1988) 229–234.

[19] S.Sahni, T.Gonzales, *P*-complete approximation problems, *J. Assoc. Comput. Mach. 23* (1976) 95–103.

[20] E.Zemel, Measuring the quality of approximate solutions to zero-one programming problems, *Math. of Operations Research 6* (1981) 319–322.