Journal of Global Optimization **27:** 83–96, 2003.
© 2003 *Kluwer Academic Publishers. Printed in the Netherlands.*

Studying the Complexity of Global Verification for NP-Hard Discrete Optimization Problems

DEREK E. ARMSTRONG[1] and SHELDON H. JACOBSON*
[1]*Alphatech, Inc., 3811 North Fairfax Drive, Arlington, VA 22203, USA;* [2]*Department of Mechanical and Industrial Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801-2906, USA (e-mail: shj@uiuc.edu)*

(Received 2 July 2001; accepted in received form 16 February 2003)

Abstract. This paper examines the complexity of global verification for MAX-SAT, MAX-*k*-SAT (for $k \geqslant 3$), Vertex Cover, and Traveling Salesman Problem. These results are obtained by adaptations of the transformations that prove such problems to be NP-complete. The class of problems PGS is defined to be those discrete optimization problems for which there exists a polynomial time algorithm such that given any solution ω, either a solution can be found with a better objective function value or it can be concluded that no such solution exists and ω is a global optimum. This paper demonstrates that if any one of MAX-SAT, MAX-*k*-SAT (for $k \geqslant 3$), Vertex Cover, or Traveling Salesman Problem are in PGS, then P=NP.

Subject classification: analysis of algorithms, computational complexity, networks/graphs, traveling salesman

Key words: computational complexity; discrete optimization problems; local search algorithms; NP-hard

1. Introduction

Discrete optimization problems in the set PGS are those problems for which there exists a polynomial time algorithm such that given a solution ω of an instance *I*, the algorithm either returns a solution ω' with better objective function value or else concludes that ω is a global optimum (Jacobson and Solow, 1993). Let P_S denote the set of search problems that can be solved in polynomial time. An interesting, unresolved question to ask is whether P_S = PGS? The research in this paper examines this question and provides evidence to support (though not prove) that this equality does indeed hold. A necessary condition for a problem to be in PGS is that global verification of any solution can be accomplished in polynomial time. This paper provides evidence that P_S and PGS are (in some sense) close by showing that for many NP-hard optimization problems, global verification is an NP-complete decision problem. Jacobson and Solow (1993) show that if there exists a PGS problem that is NP-hard, then NP = co-NP. One goal of this research is to extend this result by showing that if there exists a PGS problem that is NP-hard, then P = NP. This paper provides a step in this direction since if it can be shown that the global verification problem for every NP-hard optimization problem is NP-complete, then it is true that the existence of a PGS problem that is NP-hard implies that P = NP.

There has been a limited amount of research reported in the literature on the PGS versus P_S question and the global verification problem. Schulz et al. (1995) and Grotschel and Lovasz (1995) show that problems formulated as 0-1 Integer Programming Problems (with no restrictions on the objective function coefficients) are solvable in polynomial time if and only they are in PGS. Schulz and Weismantel (1999) also show similar results for general integer programming problems for which the independent variables are bounded by a constant. Papadimitriou and Steilglitz (1977) examine the complexity of global verification for the Traveling Salesman Problem (TSP) by showing that global verification for the TSP is NP-complete. They obtain this result by proving a particular subclass of Hamiltonian Circuit to be NP-complete and then polynomially transform this problem to the global verification problem of TSP. This paper extends these results by also showing that global verification for MAX-SAT, MAX-k-SAT (for $k \geqslant 2$), and Vertex Cover are all NP-complete. This paper also provides an alternative proof of the result that global verification for TSP is NP-complete. Furthermore, global verification for TSP is shown to be NP-complete by using a similar method used to show that the TSP decision problem is NP-complete (as given in Garey and Johnson, 1979). Therefore, the proof presented here to demonstrate that global verification for TSP is NP-complete is a simplification of the proof presented in Papadimitriou and Steilglitz (1977). To obtain all these results, the global verification problem for MAX-SAT is first shown to be NP-complete. The other global verification problems are then shown to be NP-complete by transforming the corresponding global verification problems in a similar manner as the original problems were transformed (as described in Garey and Johnson, 1979). For example, the global verification of MAX-3-SAT is proven to be NP-complete by showing the global verification of MAX-SAT polynomially transforms to the global verification problem of MAX-3-SAT, where this transformation is an adaptation of the transformation from SAT to 3-SAT (as described in Garey and Johnson, 1979).

Other related results in the literature include the introduction of the class of problems PLS (Johnson et al., 1988) along with the investigation of the complexity of finding local optima for discrete optimization problems. The class of problems PLS contains all local search problems (discrete optimization problems with a neighborhood function) in which local optimality can be verified in polynomial time. Fischer et al. (1997) discuss witness-isomorphic reductions and demonstrate how they can be used to preserve the length of augmenting paths between local search problems. An augmenting path for a local search algorithm is a sequence of solutions such that each solution in the sequence has better objective function value than the preceding solution and is in the neighborhood of the preceding solution. These witness-isomorphic reductions show how a local search algorithm for an NP-hard optimization problem can also be used to address other NP-hard optimization problems (via a witness-isomorphic reduction). Jacobson and Solow (1993) introduce the class of problems PGS and investigate finite improvement algorithms within this class. Finite improvement algorithms are iterative algorithms that terminate in a finite number of iterations, where each such iteration is performed in polynomial time and strict improvement is required from iteration to iteration. Note that PGS and PLS are equivalent in that any problem Π in PGS or PLS can be reformulated (in polynomial time in the length of problem instances of Π) as an equivalent problem in the other class.

The paper is organized as follows: Section 2 provides formal definitions and background material needed to obtain the global verification results presented in Section 3. The class of problems PGS and optimization problems MAX-k-SAT (for $k \geqslant 2$), TSP, Vertex Cover, and Hamiltonian Circuit are all formally defined in Section 2. Section 3 shows that global verification for MAX-SAT, MAX-k-SAT (for $k \geqslant 2$), Vertex Cover, and the TSP are all NP-complete. Section 4 provides concluding comments and directions for future research.

2. Definitions and Background

Several definitions are needed to describe the results presented. A *discrete optimization problem* Π with instances denoted by (Ω, f) is formulated to find a solution $\omega \in \Omega$ that minimizes or maximizes f, where Ω is the (countable) *solution space* and $f\colon \Omega \to R$ is the *objective function*. For every instance (Ω, f) of an optimization problem Π, a *neighborhood function* $\eta : \Omega \to 2^{\Omega}$ maps each solution $\omega \in \Omega$ into a subset of the solution space. For a minimization problem, a solution $\omega \in \Omega$ is a (*strict*) *local optimum* if $f(\omega)$ $(<) \leqslant f(\omega)$ for all $\omega \in \eta$ (ω), and a solution $\omega \in \Omega$ is a *global optimum* if $f(\omega) \leqslant f(\omega)$ for all $\omega \in \Omega$. Similarly, for a maximization problem, a solution $\omega \in \Omega$ is a (*strict*) *local optimum* if $f(\omega)$ $(>) \geqslant f(\omega)$ for all $\omega \in \eta(\omega)$, and a solution $\omega \in \Omega$ is a *global optimum* if $f(\omega) \geqslant f(\omega)$ for all $\omega \in \Omega$.

To define the concept of polynomial-time solvability of a discrete optimization problem, several definitions are needed (for additional details, see Garey and Johnson, 1979). A deterministic Turing machine (DTM) is a model of a computer that is used to formally define the class of problems P. A discrete optimization problem Π can be defined as a relation R over $\{0, 1\}^* \times \{0, 1\}^*$ by having $(x, y) \in R$ if and only if y is the optimal solution to the instance x of Π. Given $(x, y) \in R$, the strings of bits x and y are used to represent the instance (Ω, f) of Π and an optimal solution of (Ω, f), respectively. A function $h\colon\{0, 1\}^* \to \{0, 1\}^*$ *realizes* the string relation R if and only if for each instance x of Π, $h(x)$ equals $y \in \{0,1^*\}$ such that $(x, y) \in R$. The relation R (or optimization problem Π) is solvable in polynomial-time if there exists a polynomial-time DTM program M such that h_M realizes R. Let P_S denote the set of all discrete optimization problems that are solvable in polynomial-time. Note that this definition is equivalent to the definition of the class of search problems given in Johnson et al. (1988).

From Jacobson and Solow (1993), a *polynomial-time global search (PGS)* problem Π is an optimization problem for which the following three polynomial time algorithms exist:

1. Algorithm A_Π, given an instance (Ω, f), produces a solution $\omega \in \Omega$.
2. Algorithm B_Π, given an instance (Ω, f) and a solution $\omega \in \Omega$, computes $f(\omega)$.
3. Algorithm C_Π, given an instance (Ω, f) and a solution $\omega \in \Omega$, either outputs a new solution $\omega' \in \Omega$ with $f(\omega') < f(\omega)$ (assuming a minimization problem) or else concludes that no such solution exists and ω is a global optimum.

The first algorithm A_Π ensures that given any instance (Ω, f), a solution can be produced in polynomial time. The second algorithm B_Π ensures that the objective function can be computed in polynomial time. The third algorithm C_Π provides a procedure such that given any solution ω, the algorithm either finds a better solution or concludes that no such solution exists and ω is a global optimum. Therefore, the third algorithm C_Π is equivalent to the existence of a neighborhood function that can be searched in polynomial time and results in all the local optima being global optima.

Several decision problems can be defined from a particular optimization problem. For an optimization problem Π, let Π(D) be the decision problem:

> Given an instance (Ω, f) of Π and a real number K, does there exist a solution $\omega \in \Omega$ such that $f(\omega) \leqslant K$?

Given an optimization (minimization) problem Π with a corresponding solution $\omega \in \Omega$, the global verification decision problem Π(GV) can be defined:

> Given an instance (Ω, f) of Π and a solution $\omega \in \Omega$, does there exist a solution $\omega' \in \Omega$ such that $f(\omega') < f(\omega)$?

For example, MAX-3-SAT(GV) denotes the global verification decision problem associated with the MAX-3-SAT optimization problem.

This paper shows that the global verification problem is NP-complete for several well-known discrete optimization problems, namely MAX-SAT, MAX-*k*-SAT ($k \geqslant 2$), Vertex Cover, and the TSP. These discrete optimization problems are now formally defined. To define MAX-*k*-SAT, the following terminology and notation are needed. A *Boolean variable* is a variable that takes on only one of two values, true or false. Given a collection of Boolean variables $X = \{x_1, x_2, \dots, x_n\}$, for each $x \in X$, define $\mathbf{x}$ and $\bar{\mathbf{x}}$ to be *literals* of X, where the literal $\mathbf{x}$ ($\bar{\mathbf{x}}$) is true (true) if and only if the variable x is set to true (false). For simplification, throughout the remainder of this paper, let zero (0) denote false and one (1) denote true. A *clause* is a collection of literals of X. For example, if $X = \{x_1, x_2, x_3, x_4, x_5\}$, then $(\mathbf{x_2}, \bar{\mathbf{x}}_\mathbf{4}, \mathbf{x_5})$ is a clause. A clause is *satisfied* if any of the literals it contains is true. MAX-*k*-SAT (for $k \geqslant 1$) is now formally defined.

MAX-*k*-SAT: Given *m* clauses, where each clause consists of *k* literals, over the set of Boolean variables $X = \{x_1, x_2, \dots, x_n\}$, find a truth assignment *t*: $X \to \{0, 1\}$ that maximizes the number of satisfied clauses.

MAX-*k*-SAT has a polynomial number of distinct objective function values (i.e., integer values between 0 and *m*). This property can be used to show that MAX-*k*-SAT is polynomially solvable if and only if MAX-*k*-SAT is in PGS. Note that MAX-SAT is the same as MAX-*k*-SAT, except that any two clauses for MAX-SAT may have a different number of literals.

The TSP, Hamiltonian Circuit, and Vertex Cover are all formally defined as optimization problems. Note that the global verification problem for each of these problems is well-defined.

Traveling Salesman Problem (TSP): Given a collection of *n* cities $\{x_1, x_2, \dots, x_n\}$ and distances $d(x_i, x_j)$ for each pair of distinct cities x_i and x_j, find a Hamiltonian circuit (i.e., a permutation of the *n* cities $y_1 y_2 \dots y_n$) with minimum total length

$$\left(d(y_1, y_n) + \sum_{i=1}^{n-1} d(y_i, y_{i+1})\right).$$

Hamiltonian Circuit: Let G = (V, E) be a graph, where |V| = *n*. Find an ordering of the vertices $\{v_1, v_2, \dots, v_n\}$ such that $(v_i, v_{i+1}) \in$ E for all $i = 1, 2, \dots, p - 1$, $(v_n, v_1) \in E$, and *p* is maximized.

Vertex Cover: Let G = (V, E) be a graph. Find a subset V′ ⊆ V, with the fewest number of vertices, such that for each edge (*u*, *v*) ∈ E, either *u* ∈ V′ or *v* ∈ V′.

3. Polynomial Transformations that Preserve the PGS Property

This section shows that the global verification decision problem for several NP-hard discrete optimization problems is NP-complete. An implication of this result is that if one of these problems is in PGS, then P = NP.

Theorem 1 shows the problem of determining if there exists a truth assignment that satisfies all of the clauses, when there is a truth assignment that satisfies all but *l* clauses (i.e., a special case of MAX-SAT(GV)), is NP-complete. This special case is referred to as MAX-SAT(GV, *l*). Note that MAX-*k*-SAT(GV, *l*) is defined in a similar way, where the only difference is that all the clauses have *k* literals. The result in Theorem 1 implies that the global verification problem for MAX-SAT is NP-complete, hence if MAX-SAT is in PGS, then P = NP.

MAX-SAT(GV, *l*): Given *m* clauses over the set of Boolean variables $\{X = x_1, x_2, \dots, x_n\}$ and a truth assignment $t : X \to \{0, 1\}$ that satisfies $m - l$ clauses, does there exist a truth assignment that satisfies all *m* clauses?

THEOREM 1. *MAX-SAT(GV, l) is NP-complete for all* $l \geqslant 1$.

Proof. Let $l \geqslant 1$. This results is obtained by showing that SAT polynomially transforms to MAX-SAT(GV,*l*). First, it is straightforward to see that MAX-SAT(GV,*l*) is in NP. Let C be a set of m clauses over a set $X = \{x_1, x_2, \dots, x_n\}$ of n Boolean variables; this defines an instance of SAT. An instance of MAX-SAT(GV, *l*) can be constructed with the property that it is satisfiable if and only if C is satisfiable. To see this, let = $\bar{C} = \{(y_1, y_2, \dots, y_k, x_{n+1}, x_{n+2}, \dots, x_{n+l}\})$: $(y_1, y_2, \dots, y_k) \in C\}$. The instance of MAX-SAT(GV, *l*) is given by the set of clauses $C' = \bar{C} \cup \{\bar{x}_{n+1}, (\bar{x}_{n+2}), \dots, (\bar{x}_{n+1})\}$, set of Boolean variables $X' = X \cup \{x_{n+1}, x_{n+2}, \dots, x_{n+l}\}$, and the truth assignment $t' : X' \to \{0, 1\}$, where $t'(x_i) = 0$ for all $i = 1, 2, \dots, n$ and $t'(x_{n+1}) = 1$ for all $i = 1, 2, \dots, l$. Note that this instance of MAX-SAT(GV, *l*) is constructed in polynomial time in the length of the SAT instance. By design, the truth assignment t' does not satisfy exactly l clauses of C′. To complete the proof, it is necessary to show that C′ is satisfiable if and only if C is satisfiable. Suppose C is satisfiable and $t : X \to \{0, 1\}$ is a truth assignment that satisfies all the clauses of C. Then, it is straightforward to see that the truth assignment $t^* : X' \to \{0, 1\}$ defined by $t^*(x) = t(x)$ for all $x \in X$, and $t^*(x) = 0$ for all $x \in X' - X$, satisfies all the clauses of C′. Now, suppose C' is satisfiable and $t : X' \to \{0, 1\}$ satisfies all the clauses of C′. Since C′ is satisfiable by t, it then follows that $t(x) = 0$ for all $x \in X' - X$, and $t^* : X \to \{0, 1\}$, where $t^*(x) = t(x)$ for all $x \in X$, satisfies all the clauses in C. This implies that SAT polynomially transforms to MAX-SAT(GV, *l*).

Cook (1971) proved that SAT was NP-complete, hence established the first decision problem to be NP-complete. After SAT was shown to be NP-complete, other decision problems Π could be shown to be NP-complete by polynomially transforming SAT to Π. In a similar fashion, the work presented here first shows that the global verification problem of MAX-SAT is NP-complete. Given this result, other global verification problems Π(GV) could be shown to be NP-complete by polynomially transforming MAX-SAT(GV) to Π(GV). □

Theorem 2 shows that the global verification problem for MAX-3-SAT is also NP-complete. The polynomial transformation used in the proof of Theorem 2 is the same transformation to show SAT polynomially reduces to 3-SAT (as described in Garey and Johnson, 1979), except for changes that are needed to transform a truth assignment of SAT to a truth assignment of 3-SAT.

THEOREM 2. *MAX-3-SAT(GV, l) is NP-complete for all* $l \geqslant 1$.

Proof. Let $l \geqslant 1$. First, it is straightforward to see that MAX-3-SAT(GV, *l*) is in NP. The proof follows by showing that MAX-SAT(GV, *l*) polynomially transforms to MAX-3-SAT(GV, *l*). Let C = $\{c_1, c_2, \dots, c_m\}$ be a set of m clauses over a set $X = \{x_1, x_2, \dots, x_n\}$ of n Boolean variables and $t : X \to \{0, 1\}$ be a truth assignment that satisfies all but l clauses; this defines an instance of MAX-SAT(GV,*l*). A set of clauses C′, where each clause in C′ has exactly three literals over a set X' of Boolean variables, and a truth assignment $t' : X' \to \{0, 1\}$ that satisfies all but l clauses of C′ can be constructed such that C is satisfiable if and only if C′ is satisfiable. For each clause in C, additional variables and clauses are needed to construct the sets C′ and X'. Let $c_j = (y_1, y_2, \dots, y_k) \in$ C be a clause. Consider the following four cases based on the size of k.

Case 1: $k = 1$. Let $X_j = \{w_{1,j}, w_{2,j}\}$ and $\mathrm{C}_j = \{(y_1, w_{1,j}, w_{2,j}), (y_1, \bar{w}_{1,j}, w_{2,j}), (y_1, w_{1,j}, \bar{w}_{2,j}), (y_1, \bar{w}_{1,j}, \bar{w}_{2,j})\}$.
Case 2: $k = 2$. Let $X_j = \{w_{1,j}\}$ and $C_j = \{(y_1, y_2, w_{1,j}), (y_1, y_2, \bar{w}_{1,j})\}$.
Case 3: $k = 3$. Let $X_j = \phi$ and $\mathrm{C}_j = \{(c_j)\}$.
Case 4: $k > 3$. Let $X_j = \{w_{i,j} : 1 \leqslant i \leqslant k-3\}$ and
$\mathrm{C}_j = \{(y_1, y_2, w_{1,j})\} \cup \{(\bar{w}_{i,j}, y_{i+2}, w_{i+1,j}) : i = 1, 2, \dots, k-4\}$
$\cup\{\bar{w}_{k-3,j}, y_{k-1}, y_k)\}$.

Let

$$C' = \bigcup_{j=1}^{m} C_j \text{ and } X' = X \cup \left[\bigcup_{j=1}^{m} X_j\right].$$

A truth assignment $t' : X \to \{0, 1\}$ is defined that satisfies all but l clauses of C′. For each $x \in X$, set $t'(x) = t(x)$. Let $c_j = (y_1, y_2, \dots, y_k) \in$ C (where $k > 3$) be a clause that is not satisfied by t. By setting $t'(x) = 1$, for each $x \in X_j$, it follows that t' satisfies all but one clause of C_j. Now, let $c_j = (y_1, y_2, \dots, y_k) \in$ C (where $k > 3$) be a clause that is satisfied by t. Let p be the smallest integer such that the literal y_p is satisfied by t. The following rules are used to define $t'(x)$ for all $x \in X_j$:

(1)if $p = 1$ or 2, then $t'(w_{i,j}) = 0$ for all $i = 1, 2, \dots, k-3$,
(2)if $p = k-1$ or k, then $t'(w_{i,j}) = 1$ for all $i = 1, 2, \dots, k-3$,
(3)if $2 < p < k-1$, then $t'(w_{i,j}) = 1$ for all $i = 1, 2, \dots, p-2$, and
$t'(w_{i,j}) = 0$ for all $i = p-1, p, \dots, k-3$.

Therefore, t' satisfies all clauses in C_j. Finally, for each j such that $c_j = (y_1, y_2, \dots, y_k)$, where $k \leqslant 3$, set $t'(x) = 1$ for all $x \in X_j$. In this case, if t satisfies c_j, then t' satisfies all the clauses in C_j. Moreover, if t does not satisfy c_j, then t' satisfies all but one clause of C_j. By construction, the truth assignment t' satisfies all but l clauses of C′. Lastly, C is satisfiable if and only if C′ is satisfiable (Garey and Johnson, 1979). Note that the constructions of C′, X', and t' are all obtained in a polynomial number of steps in the length of instances of MAX-SAT(GV, l). Therefore, MAX-3-SAT(GV, l) is NP-complete. □

Theorem 3 extends the result in Theorem 2 to show that MAX-k-SAT(GV, l) is NP-complete for all $k \geqslant 3$. MAX-2-SAT(GV,l) is polynomially solvable for all $l \geqslant 1$ since 2-SAT is polynomially solvable (Aspvall et al., 1979). However, the optimization problem MAX-2-SAT is NP-hard. Theorem 4 proves that MAX-2-SAT(GV) is NP-complete.

THEOREM 3. *MAX-k-SAT(GV, l) is NP-complete for all k⩾3 for all l⩾1.*

Proof. The proof is by induction on *k*. Let $l \geqslant 1$. The base case (*k* = 3) follows from Theorem 2. Suppose MAX-*k*-SAT(GV, *l*) is NP-complete for some $k \geqslant 3$. It will be shown that MAX-*k*+1-SAT(GV, *l*) is NP-complete. Let C = $\{c_1, c_2, \dots, c_m\}$ be a set of *m* clauses over the set $X = \{x_1, x_2, \dots, x_n\}$ of *n* Boolean variables, and let $t : X \to \{0, 1\}$ be a truth assignment that satisfies all but *l* clauses. This defines an instance of MAX-*k*-SAT(GV, *l*). For each $c_j = (y_1, y_2, \dots, y_k) \in$ C, let $X_j = \{w_j\}$ and $\mathrm{C}_j = \{(y_1, y_2, \dots, y_k, \bar{w}_j\}), (y_1, y_2, \dots, y_k, w_j)\}$. Let

$$\mathrm{C}' = \bigcup_{j=1}^{m} C_j \text{ and } X' = X \cup \left[\bigcup_{j=1}^{m} X_j\right]$$

Define $t' : X' \to \{0, 1\}$ by $t'(x) = t(x)$ for all $x \in X$, and $t'(x) = 1$, for all $x \in X' - X$. If t satisfies clause c_j, then t' satisfies both clauses in C_j. If t does not satisfy clause c_j, then t' satisfies only one clause in C_j. Since t satisfies all but l clauses of C, it then follows that t' satisfies all but l clauses of C′. Therefore, C′, X', and t' define an instance of MAX-*k*+1-SAT(GV, *l*) that can be constructed in polynomial time (in the length of instances of MAX-*k*-SAT(GV, *l*)). By their definition, C is satisfiable if and only if C′ is satisfiable. Therefore, MAX-*k*-SAT(GV, *l*) polynomially transforms to MAX-*k*+1-SAT(GV, *l*), hence by induction, MAX-*k*-SAT(GV, *l*) is NP-complete for all $k \geqslant 3$. □

The proof of Theorem 4 follows from an extension of the transformation from 3-SAT to MAX-2-SAT(D) given in Garey et al. (1976).

THEOREM 4. *MAX-2-SAT(GV) is NP-complete.*

Proof. First, it is straightforward to see that MAX-2-SAT(GV) is in NP. The proof follows by showing that MAX-3-SAT(GV, 1) polynomially transforms to MAX-2-SAT(GV). Let C = $\{c_1, c_2, \dots, c_m\}$ be a set of *m* clauses over the set $X = \{x_1, x_2, \dots, x_n\}$ of *n* Boolean variables and let $t : X \to \{0, 1\}$ be a truth assignment that satisfies all but one clause; this defines an instance of MAX-3-SAT(GV, 1). Let $X' = X \cup \{a_i, b_i, c_i, d_i, p_i : i = 1, 2, \dots, m\}$. For each clause $c_j = (y_1, y_2, y_3) \in$ C, let $C_j = \{(y_1, a_j), (y_1, \bar{a}_j), (y_2, b_j), (y_2, \bar{b}_j), (y_3, c_j), (y_3, \bar{c}_j), (\bar{y}_1, \bar{y}_2), (\bar{y}_2, \bar{y}_3), (\bar{y}_1, \bar{y}_3), (y_1, \bar{d}_j), (y_2, \bar{d}_j), (y_3, \bar{d}_j), (d_j, p_j), (d_j, \bar{p}_j)\}$. It is straightforward to verify that if clause c_j is satisfied, then exactly 11 clauses in C'_j can be satisfied. If the clause c_j is not satisfied, then exactly 10 clauses in C'_j can be satisfied. Let

$$C' = \bigcup_{j=1}^{m} C'_j.$$

Then there exists a truth assignment that satisfies exactly $11m$ or more clauses of C' if and only if C is satisfied. It is straightforward to use the truth assignment t to construct a truth assignment $t' : X' \rightarrow \{0, 1\}$ that satisfies $11m-1$ clauses of C'. Therefore, MAX-3-SAT(GV, 1) polynomially transforms to MAX-2-SAT(GV) and by Theorem 2, it follows that MAX-2-SAT(GV) is NP-complete. □

Theorem 5 shows how the polynomial transformation from 3-SAT to Vertex Cover (Garey and Johnson, 1979) can be used to show that MAX-3-SAT(GV, *l*) polynomially reduces to a global verification problem version of Vertex Cover.

Vertex Cover(GV, *l*): Let G = (V, E) be a graph and m, l be positive integers. Let V′ be a vertex cover for G that contains $m + l$ vertices, does G have a vertex cover with m or fewer vertices?

THEOREM 5. *Vertex Cover(GV, l) is NP-complete for all $l \geqslant 1$.*

Proof. Let $l \geqslant 1$. Clearly, Vertex Cover(GV, *l*) is in NP. The proof follows by showing that MAX-3-SAT(GV, *l*) polynomially transforms to Vertex Cover(GV, *l*). Let $C = \{c_1, c_2, \ldots, c_m\}$ be a set of m clauses over the set $X = \{x_1, x_2, \ldots, x_n\}$ of n Boolean variables and $t : X \rightarrow \{0, 1\}$ be a truth assignment that satisfies all but l clauses; this defines an instance of MAX-3-SAT(GV,*l*) that can be polynomially transformed to an instance of Vertex Cover(GV, *l*). To see thus, the set of clauses C and Boolean variables X are transformed in the same manner used to prove that 3-SAT polynomially transforms to Vertex Cover (Garey and Johnson, 1979). The truth assignment t is then transformed to a vertex cover of the corresponding graph. For each variable $x_i \in X$, let $V_i = \{x_i, \bar{x}_i\}$ and $E_i = \{(x_i, \bar{x}_i)\}$. For each clause $c_j = (y_{1j}, y_{2j}, y_{3j})$, let $V'_j = \{a_{1j}, a_{2j}, a_{3j}\}$, $E'_j = \{(a_{1j}, a_{2j}), (a_{1j}, a_{3j}), (a_{2j}, a_{3j})\}$, and $E''_j = \{(a_{1j}, y_{1j}), (a_{2j}, y_{2j}), (a_{3j}, y_{3j})\}$. Now, define G=(V, E) where

$$V = \left(\bigcup_{i=1}^{n} V_i\right) \cup \left(\bigcup_{j=1}^{m} V'_j\right) \text{ and } E = \left(\bigcup_{i=1}^{n} E_i\right) \cup \left(\bigcup_{j=1}^{m} E'_j\right) \cup \left(\bigcup_{j=1}^{m} E''_j\right).$$

The graph G has a vertex cover of size less than or equal to $n + 2m$ if and only if C is satisfiable (Garey and Johnson, 1979). The truth assignment t is transformed to a vertex cover V′ of G. If $t(x_i) = 1(0)$, then $x_i(\bar{x}_i) \in V'$. Suppose c_j is a clause satisfied by t. Then one of the literals y_{ij} is true, which implies that the edge (a_{ij}, y_{ij}) is covered, since $y_{ij} \in V'$. Now, let the vertices $a_{pj} \in V'_j, a_{pj} \neq a_{ij}$, be in V′. If c_k is a clause not satisfied by t, then let each vertex in V'_k be in V′. It is straightforward to show that V′ is a vertex cover of G, with $n + 2(m - l) + 3l = n + 2m + l$ vertices. The transformation of t to the vertex cover V′ is done in polynomial time in the length of the MAX-3-SAT(GV, *l*) instances. Therefore, MAX-3-SAT(GV, l) polynomially transforms to Vertex Cover(GV, l), hence by Theorem 2, Vertex Cover(GV, l) is NP-complete. $\square$

Theorem 6 implies that given a graph and a Hamiltonian path, the problem of determining if the graph has a Hamiltonian Circuit is still NP-complete. Papadimitriou and Steilglitz (1977) establish the same result. This result here is obtained by an alternative and independent proof. First, Hamiltonian Circuit(GV, l) is formally stated.

Hamiltonian Circuit(GV, l). Let G = (V, E) be a graph, where $|\mathrm{V}| = n$, and $\{v_1, v_2, \dots, v_n\}$ is an ordering of the vertices such that $(v_{i+1}, v_i + 1) \in \mathrm{E}$ for all $i = 1, 2, \dots, n - l$. Does there exist an ordering of the vertices $\{u_1, u_2, \dots, u_n\}$ such that $(u_i, u_{i+1}) \in \mathrm{E}$ for all $i = 1, 2, \dots, n - 1$ and $(u_n, u_1) \in \mathrm{E}$?

Theorem 6 shows that Hamiltonian Circuit(GV, l) is NP-complete by a slight modification of the polynomial transformation used to prove that Hamiltonian circuit is NP-complete (Garey and Johnson 1979).

THEOREM 6. *Hamiltonian Circuit(GV, l) is NP-complete for all $l \geqslant 1$.*

Proof. Let $l \geqslant 1$. First, it is straightforward to see that Hamiltonian Circuit(GV, l) is in NP. The proof follows by showing that MAX-3-SAT(GV, l) polynomially transforms to Hamiltonian Circuit(GV, l). Let $\mathrm{C} = \{c_1, c_2, \dots, c_m\}$ be a set of m clauses over a set $X = \{x_1, x_2, \dots, x_n\}$ of n Boolean variables and $t : X \to \{0, 1\}$ be a truth assignment that satisfies all but l clauses; this defines an instance of MAX-3-SAT(GV, l). As in the proof of Theorem 5, this instance of MAX-3-SAT(GV,l) is transformed to an instance of Vertex Cover(GV, l), where the resulting instance of Vertex Cover(GV, l) is denoted by graph G = (V, E) with vertex cover V'. By the proof of Theorem 5, $|V'| = n + 2m + l$. Let $K = n + 2m + l$. The instance of Vertex Cover(GV, l) can be transformed to an instance of Hamiltonian Circuit(GV, l). To see this, for each edge $e = (u, v) \in \mathrm{E}$, let $\mathrm{V}'_e = \{[u, e, i], [v, e, i] : i = 1, 2, \dots, 6\}$ and

$$\begin{aligned}\mathrm{E}'_e = \{&[u, e, i], [u, e, i + 1]), ([v, e, i], [v, e, i + 1]) : i = 1, 2, \dots, 5\}\\ \cup \{&([u, e, 3], [v, e, 1]), ([v, e, 3], [u, e, 1]), (u, e, 6], [v, e, 4]),\\ &([v, e, 6], [u, e, 4])\}\end{aligned}$$

For each vertex $v \in \mathrm{V}$, let the edges incident to v be given by $e_{v[1]}, e_{v[2]}, \dots ..,$ $e_{v[deg(v)]}$, where deg(v) denotes the degree of vertex v. Then for each vertex $v \in \mathrm{V}$, let $\mathrm{E}'_v = \{([v, e_{v[i]}, 6]), [v, e_{v[i+1]}, 1]) : i = 1, 2, \dots, \deg(v) - 1\}$. For each $k \geqslant 1$, let $\mathrm{V}_k = \{a_i : i = 1, 2, \dots, k\} \cup \left(\cup_{e\in\mathrm{E}}\mathrm{V}'_e\right)$ and $\mathrm{E}_k = \left(\cup_{e\in\mathrm{E}}\mathrm{E}'_e\right) \cup \left(\cup_{v\in\mathrm{V}}\mathrm{E}'_v\right) \cup \mathrm{E}''_k$, where $\mathrm{E}''_k = \{(a_i, [v, e_{v[1]}, 1]), (a_i, [v, e_{v[deg(v)]}, 6]) : i = 1, 2, \dots, k, v \in V\}$. Lastly, let G_k = (V_k, E_k). From Garey and Johnson (1979), the graph G_k has a Hamiltonian Circuit if and only if G has a vertex cover of size less than or equal to k. A graph G′ = (V′, E′), with path $v_1 v_2 \dots v_{|V'|-l+1}$ can be constructed such that G′ has a Hamiltonian Circuit if and only if G has a vertex cover with fewer than $K - l + 1$ vertices. Since G has a vertex cover of size K, the graph G_K has a Hamiltonian circuit $\Lambda = a_1 v_2 v_3 \ldots v_p a_1$. Define the graph G′ = (V′, E′) where $\mathrm{V}' = \mathrm{V}_K \cup \{b_1, b_2, \ldots, b_l\}$ and $\mathrm{E}' = \mathrm{E}_K \cup \{(b_j, a_i) : i = 1, 2, \ldots, K, j = 1, 2, \ldots, l\}$. Then $b_1 a_1 v_2 v_3 \ldots v_p$ is a path of length $|\mathrm{V}'| - l$ for graph G′. The graph $\mathrm{G}' = (\mathrm{V}', \mathrm{E}')$ and path $b_1 a_1 v_2 v_3 \ldots v_p$ define an instance of Hamiltonian Circuit(GV, *l*). Showing that G′ has a Hamiltonian circuit if and only if C is satisfiable completes the proof. To that end, suppose that $\Lambda_1 = b_1 u_1 u_2 \ldots u_p b_1$ is a Hamiltonian circuit for graph G′. Since Λ_1 is a Hamiltonian circuit, then $u_1 = a_i$ and $u_p = a_j$ for some $i \neq j$. Let $\mathrm{G}_{K-1} = (\{a_t : t = 1, 2, \ldots, K - 1, t \neq j\} \cup \left(\cup_{e \in \mathrm{E}} \mathrm{V}'_e\right), \left(\cup_{e \in \mathrm{E}} \mathrm{E}'_e\right) \cup \left(\cup_{v \in \mathrm{V}} \mathrm{E}'_v\right) \cup \mathrm{E}^m_{K-1})$ be a graph, where $\mathrm{E}''_{K-1} = \{(a_t, [v, e_{v[1]}, 1]), (a_t, [v, e_{v[\deg(v)]}, 6]) : t = 1, 2, \ldots, K - 1, t \neq j, v \in V\}$. Since the edge $(u_1, u_{p-1}) \in \mathrm{E}''_{K-1}$, then the graph G'_{K-1} has a Hamiltonian circuit $\Lambda_2 = u_1 u_2 \ldots u_{p-1} u_1$. The graph G'_{K-1} is isomorphic to the graph G_{K-1}, which implies that G_{K-1} has a Hamiltonian circuit. Writing Λ_2 as $b_2 \ldots b_2$, this process can be repeated to show that G_{K-2} has a Hamiltonian circuit. Continuing this process, it follows that G_{K-l} has a Hamiltonian circuit. Therefore, the graph G has a vertex cover of size less than or equal to $K - l = n + 2m$, which also implies that C is satisfiable. Now, suppose that C is satisfiable, then the graph G has a vertex cover of size less than or equal to $K - l$. This implies that the graph G_K has a Hamiltonian circuit $\Lambda_2 = a_1 u_2 u_3 \ldots u_{p-1} a_1$. Since the edges $\{(b_j, a_i) : i = 1, 2, \ldots, K, j = 1, 2, \ldots, l\}$ are in E′, it is straightforward to see that G′ has a Hamiltonian circuit. It then follows that Vertex Cover(GV, *l*) polynomially transforms to Hamiltonian Circuit(GV, *l*). Lastly, from Theorem 5, this implies that Hamiltonian Circuit(GV, *l*) is NP-complete. □

The global verification problem of TSP, TSP(GV, *l*) for $l \geqslant 1$, can be shown to be NP-complete. First, TSP(GV,*l*) is formally stated.

TSP(GV, *l*): Given a collection of *n* cities $\{x_1, x_2, \ldots, x_n\}$, distances $d(x_i, x_j)$ for each pair of distinct cities x_i and x_j, and a permutation of the cities $z_1 z_2 \ldots z_n$ with length δ; does there exist a permutation of the *n* cities, $y_1 y_2 \ldots y_n$, with length less than or equal to $\delta - l$.

TSP(GV, *l*) is NP-complete since the Hamiltonian Circuit problem discussed in Theorem 6 can be transformed to a corresponding instance of TSP(GV, *l*), where $d(x, y) = 1(2)$ if and only if $(x, y) \in (\notin) E$.

COROLLARY 1. *TSP(GV, l) is NP-complete.*

The results in this section show that for several NP-hard optimization problems, global verification is NP-complete. Furthermore, in many cases, the proofs that global verification is NP-complete can be obtained by simple adaptations to the proofs used to show that the decision problems themselves are NP-complete. In particular, MAX-3-SAT(GV, *l*) is shown to be NP-complete by polynomially transforming MAX-SAT(GV, *l*) to MAX-3-SAT(GV, *l*), similar to how SAT is polynomially transformed to 3-SAT. In transforming an instance of MAX-SAT(GV,*l*) to an instance of MAX-3-SAT(GV,*l*), a set of clauses C, a set of Boolean variables *X*, and a truth assignment $t : X \to \{0, 1\}$ that satisfies all but *l* clauses (which defines an instance of MAX-SAT(GV, *l*)) is polynomially transformed to an instance of MAX-3-SAT(GV,*l*). First, the set of clauses C and Boolean variables *X* are transformed to an instance of 3-SAT, given by C′ and *X*′. Then the truth assignment *t* is transformed to a truth assignment $t' : X' \to \{0, 1\}$ that satisfies all but *l* clauses of C′. Therefore, the proof that MAX-3-SAT(GV,*l*) is NP-complete is obtained by modifications to the proof used to show that 3-SAT is NP-complete. This same idea is then used to show that global verification is NP-complete for other NP-hard discrete optimization problems.

4. Conclusion and Directions of Future Research

The results in this paper support the conjecture that global verification is NP-complete for all NP-hard optimization problems. Note that if it can be shown that global verification is NP-complete for all NP-hard discrete optimization problems, then the existence of any one NP-hard problem in PGS implies that P = NP. One goal of this research is to show that the existence of a NP-hard problem in PGS implies P = NP. This would be a natural extension to the result in Jacobson and Solow (1993) that states that if there exists a PGS problem that is NP-hard, then NP = co-NP.

This paper shows that global verification is NP-complete for several NP-hard discrete optimization problems (MAX-SAT, MAX-*k*-SAT (for $k \geqslant 2$), Vertex Cover, TSP). These results may suggest a way to prove that, in fact, global verification is NP-complete for other NP-hard discrete optimization problems. Global verification in polynomial time may be one indicator that a discrete optimization problem is not hard. This research examines if global verification in polynomial time is sufficient for a discrete optimization problem to be solved in polynomial time. Note that there do exist discrete optimization problems in which global verification can be completed in polynomial time but it is not known if the problem itself can be solved in polynomial time. For example, a discrete optimization problem associated with the linear complementarity problem (LCP) with *P*-matrices has been shown to be in PGS, though it is not known whether this problem is NP-hard or solvable in polynomial time (Jacobson and Solow, 1993). LCP with *P*-matrices is specified as follows:

Given an $n \times n$ matrix M with positive principal minors and an n-vector q, find two n-vectors w and z such that $w = Mz + q$, $w, z \geqslant 0$, and $w^T z = 0$.

The associated discrete optimization problem that is in PGS (Jacobson and Solow, 1993) is to minimize θ such that $\theta e = w - Mz - q$, where e is an n-vector of all ones. The solutions to this discrete optimization problem are given by the basic feasible solutions to $w = Mz + q + \theta e$. The three polynomial algorithms A_Π, B_Π, and C_Π that are needed to show that this problem is in PGS are contained in

Lemke's algorithm (1965). Therefore, for this optimization version of LCP with *P*-matrices, global verification can be done in polynomial time, though it is not known if the problem itself can be solved in polynomial time. The work presented here and in Jacobson and Solow (1993) suggest that global verification for this problem is unlikely to be NP-complete.

Note that the notion of NP-hardness for LCP with *P*-matrices should not be taken from the perspective of standard complexity theory (Garey and Johnson, 1979). In Garey and Johnson (1979), polynomially solvable search problems are required to have polynomial time recognizable instances. Therefore, from standard complexity theory, LCP with *P*-matrices would be NP-hard due to the NP-hardness of recognizing *P*-matrices. In this paper, a problem Π is NP-hard if there exists an NP-complete problem Π' such that Π' Turing reduces to Π. That is, a problem Π is NP-hard if an (all) NP-complete problem(s) Π' can be solved by an algorithm T that uses a hypothetical subroutine S for solving Π such that, if S is a polynomial time algorithm for Π, then T is a polynomial time algorithm for Π' (Garey and Johnson, 1979). A call to the subroutine S by algorithm T implies that the subroutine S is given acceptable input (instance of Π); the subroutine S is not assumed to recognize a valid instance in polynomial time (Megiddo, 1988).

The results in this paper also imply that the existence of one NP-hard problem in PGS implies that other NP-hard problems are in PGS. Since it was proven that MAX-SAT(GV, l) is NP-complete, if MAX-SAT is in PGS, then P = NP and all problems that are NP-easy can be solved in polynomial time.

One future direction of this research is to extend these results to show that global verification is NP-complete for all NP-hard discrete optimization problems. Obtaining this result appears challenging, so a preliminary research goal is to find large sets of NP-hard discrete optimization problems such that global verification is NP-complete for every problem in that set. Also, it would be useful to find properties of problems that imply that their global verification problem is NP-complete. Answers to these issues and questions may lead to a better understanding of the P_S versus PGS question.

Acknowledgements

The authors wish to thank the Associate Editor and two anonymous referees for their feedback on the paper.

This research is supported in part by the National Science Foundation (DMI-9907980) and the Air Force Office of Scientific Research (F49620-01-1-0007).

References

Aspvall, B., Plass, M.F. and Tarjan, R.E. (1979), A linear-time algorithm for testing the truth of certain quantified Boolean formulas, *Information Processing Letters*, 8(3), 121–123.

Cook, S.A., (1971), The complexity of theorem proving procedures, *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, Association for Computing Machinery, New York, pp. 151–158.

Even, S., Itai, A. and Shamir, A. (1976) On the complexity of timetable and multicommodity flow problems, *SIAM Journal on Computing* 5, 691–703.

Fischer, S., Hemaspaandra, L. and Torenvliet, L. (1997), Witness-isomorphic reductions and local search. In: Sorbi, A. (ed), *Lecture Notes in Pure and Applied Mathematics*, Marcel Dekker, New York, NY, pp. 207–223.

Garey, M.R. and Johnson, D.S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, New York, NY.

Garey, M.R., Johnson, D.S. and Stockmeyer, L. (1976), Some simplified NP-complete graph problems, *Theoretical Computer Science* 1, 237–267.

Grotschel, M. and Lovasz, L. (1995), Combinatorial Optimization. In: R. Graham, M. Grotschel, L. Lovasz (eds), *Handbook of Combinatorics*, vol. II (North-Holland, Amsterdam), pp. 1541–1597.

Jacobson, S.H. and Solow, D. (1993), The effectiveness of finite improvement algorithms for finding global optima, *Methods and Models of Operations Research*, 37, 257–272.

Johnson, D.S., Papadimitriou, C.H. and Yannakakis, M. (1988), How easy is local search? *Journal of Computer and System Sciences* 37, 79–100.

Lemke, C.E. (1965), Bimatrix equilibrium points and mathematical programming, *Management Science* 11, 442–455.

Megiddo, N. (1988), A note on the complexity of P-matrix LCP and computing an equilibrium, Research Report RJ 6439, IBM Almaden Research Center, San Jose, CA.

Papadimitriou, C.H. and Steilglitz, K. (1977), On the complexity of local search for the traveling salesman problem, *SIAM Journal on Computing* 6, 76–83.

Schulz, A.S. and Weismantel, R. (1999), An oracle-polynomial time augmentation algorithm for integer programming, *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, Baltimore, MD, pp. 967–968.

Schulz, A.S., Weismantel, R. and Ziegler, G.M. (1995), 0/1-Integer programming: optimization and augmentation are equivalent. In: P. Spirakis, (ed), *Lecture Notes in Computer Science*, no. 979, Springer, Berlin, pp. 473–483.