Finding Multiple Optimal Solutions to an Integer Linear Program by Random Perturbations of its Objective Function

Noah Schulhof , Pattara Sukprasert , Eytan Ruppin , Samir Khuller , Alejandro A. Schaffer *

Posted Date: 21 January 2025

doi: 10.20944/preprints202501.1518.v1

Keywords: integer linear program; selection problems; multiple optima; randomized algorithms; parallel algorithms

Article

Finding Multiple Optimal Solutions to an Integer Linear Program by Random Perturbations of Its Objective Function

Noah Schulhof [1], Pattara Sukprasert [2], Eytan Ruppin [3], Samir Khuller [4] and Alejandro A. Schäffer [5,*]

[1] Cancer Data Science Laboratory, National Cancer Institute, National Institutes of Health, Bethesda, MD USA and Department of Computer Science, Northwestern University, Evanston, IL USA; nschulhof@u.northwestern.edu
[2] Databricks Inc, San Francisco, CA USA; pattara.sk127@gmail.com
[3] Department of Computer Science, Northwestern University, Evanston, IL USA; samir.khuller@northwestern.edu
[4] Cancer Data Science Laboratory, National Cancer Institute, National Institutes of Health, Bethesda, MD USA; eytan.ruppin@nih.gov
[5] Cancer Data Science Laboratory, National Cancer Institute, National Institutes of Health, Bethesda, MD USA; alejandro.schaffer@nih.gov
* Correspondence: alejandro.schaffer@nih.gov

Abstract: Integer linear programs (ILPs) and mixed integer programs (MIPs) often have multiple distinct optimal solutions, yet the widely-used Gurobi optimization solver returns certain solutions at disproportionately high frequencies. This behavior is disadvantageous, as, in fields such as biomedicine, the identification and analysis of distinct optima yields valuable domain-specific insights that inform future research directions. In the present work, we introduce MORSE (Multiple Optima via Random Sampling and careful choice of the parameter Epsilon), a randomized, parallelizable algorithm to efficiently generate multiple optima for ILPs. MORSE maps multiplicative perturbations to the coefficients in an instance's objective function, generating a modified instance that retains an optimum of the original problem. We formalize and prove the above claim in some practical conditions. Furthermore, we prove that for 0/1 selection problems, MORSE finds each distinct optimum with equal probability. We evaluate MORSE using two measures; the number of distinct optima found in r independent runs, and the diversity of the list (with repetitions) of solutions by average pairwise Hamming distance and Shannon entropy. Using these metrics, we provide empirical results demonstrating that MORSE outperforms the Gurobi method and unweighted variations of the MORSE method on a set of 20 Mixed Integer Programming Library (MIPLIB) instances and on a combinatorial optimization problem in cancer genomics.

Keywords: integer linear program; selection problems; multiple optima; randomized algorithms; parallel algorithms

1. Introduction

An integer linear program (ILP) or a mixed integer program (MIP) may have multiple distinct optimal solutions. Typical usage of ILP solvers, such as Gurobi, SCIP, and CPLEX, finds the optimal value of the objective function and one optimal solution. In the present work, we implement and test MORSE (Multiple Optima via Random Sampling and careful choice of the parameter Epsilon), an efficient and parallelizable algorithm to generate distinct optima for ILPs and MIPs with multiple optimal solutions that differ in at least one of the variables constrained to be integral. An example of a necessarily linear objective function for an ILP is $3.2x + 1.5y$. The variables x and y must take on integer values, but the coefficients 3.2 and 1.5 and the value of the objective function need not be integers. The MORSE algorithm maps multiplicative perturbations to the coefficients on binary and integer variables in the objective function, while keeping all model constraints unchanged. The intuition is

 doi:10.20944/preprints202501.1518.v1

that the random perturbation breaks ties among the original optimal solutions, determining which solution(s) are favored in the perturbed problem.

1.1. Problem Motivation

This project was primarily motivated by a problem in cancer genomics for which we implemented the software MadHitter [1], which uses Gurobi or SCIP to solve variations of the classical combinatorial optimization problem *minimum hitting set* [2]. Taking as input single-cell mRNA data from patient tumors, MadHitter finds an optimal combination of target genes to treat either an individual patient or a cohort of multiple patients, and suggests different genes and their encoded proteins as future targets for cancer drug development. For the modeling in MadHitter, we treat each targetable gene equivalently, such that any solution targeting g genes is advantageous to any solution targeting $g+1$ genes, regardless of what may be known about the genes. MadHitter does not choose between two optimal gene sets of the same minimum size. Clinically, most genes cannot be targeted by existing drugs, suggesting that solutions with more "druggable" genes are preferable. Moreover, biochemists and drug developers have expert knowledge as to which genes will be easier to target chemically with future drugs. Therefore, they are interested in identifying multiple optimal solutions to choose a solution containing genes that may be targeted for the treatment of future patients.

Gurobi offers a `solution pool` method that finds multiple optimal solutions of a linear program. Technically, if `model` stores the Gurobi problem instance, then one specifies the intent to collect multiple solutions with a command such as:

```
model.setParam(GRB.param.PoolSolutions, num_solutions)
```

and then one retrieves the i^{th} solution with a command such as:

```
select_solution(model, i).
```

We tested the `solution pool` method within MadHitter, but regrettably found that many solutions were frequently missed when sampling 50-100 optimal solutions. Many of the optima returned were 'copies' of the same optimal hitting sets, differing only in the values of decision variables appearing exclusively in the model constraints. However, only solutions that differ in the objective function variable values are of interest for MadHitter, as well as in most practical optimization problems. The difficulties in using `solution pool` motivated us to design and implement a novel method, MORSE, to generate distinct optimal solutions.

There are two available methods that aim to find all optimal solutions of integer programs. Both methods modify the use of the branch-and-bound tree in finding a single optimal solution. The unrestricted subtree method is implemented via the functions `SCIPcount` and `SCIPgetNCountedSols` in SCIP [3]; the first function finds the distinct solutions, and the second function returns the number of distinct solutions found. The SCIP documentation provides more information on these functions. The one tree method is implemented via the function `populate` in CPLEX [4]. Associated with the function is a parameter denoted by either `CPX_PARAM_MIP_POOL_CAPACITY`, `SolnPoolCapacity`, or `PopulateLim` (depending on the programming language interface) that sets an upper bound on the number of solutions stored. The function `populate` allows the user to specify that suboptimal solutions within some percentage of the optimal objective function value should also be returned. For more information, see the CPLEX documentation. Both methods are inherently sequential, and the solutions are unavailable until the respective functions have fully completed their execution. Both methods are yet to be implemented in Gurobi as indicated by the current Gurobi documentation (accessed 12/31/24).

Closely related works seek to identify a set of k diverse solutions that are optimal (or near-optimal) and maximize some measure of diversity. The most popular measure of diversity is the sum of the Hamming distances between the values of the integer variables [5]. We explain the Hamming distance measure and an alternative in Section 2. Danna and Woodruff [5] formulated the problem of finding k diverse near-optimal solutions, executing the CPLEX one-tree method for a fixed duration to generate a large set S of near-optima. From the set S, the [5] exact method selects the most diverse set of k solutions. Danna and Woodruff also proposed a local search heuristic method for instances for which

 doi:10.20944/preprints202501.1518.v1

the aforementioned method is excessively slow [5]. Trapp and Konrad [6] proposed a different method to find k optima for 0-1 ILPs. This method produces solutions one at a time, such that solution $s+1$ is maximally diverse with respect to the previously produced s solutions for $s \geq 1$. An advantage of this sequential method is that one can examine solutions one at a time and decide whether to continue. A disadvantage is that the method does not handle general integer variables. In sum, two gaps in the previous work are that Gurobi lacks a method to find all optimal solutions, and the previous methods in SCIP and CPLEX are inherently sequential. In the present work, we tested primarily with Gurobi and secondarily with SCIP and CPLEX.

1.2. Additional Literature Review

In biomedicine, including cancer genomics, researchers using ILPs have repeatedly been interested in finding multiple optimal solutions [7–9]. In biology, understanding the diversity of optimal solutions can yield valuable insights [7]. For some biological problems, there is specialized knowledge that renders certain solutions preferable to others for reasons that are not represented in the objective function [10,11]. Biomedical papers (some describing named scientific software packages) that describe methods to find multiple optima for ILPs or MIPs include: MedØlDatschgerl by [12], [8,13], MTM by [9], [14] CellNetAnalyzer by [15], Rosetta by [16], OptCouple by [17], [18], Corex by [10], DEXOM by [19], [20,21], OCSANA by [22], and MCSEnumerator by [11]. Several studies also exist on non-biomedical topics including energy management [23], materials science [24], computer chip layout [25], user interface design [26], urban planning [27] and clustering [28] that attempted to enumerate all optima for similar reasons.

Because the problem of finding the cost of any optimal solution to ILPs is NP-complete [29, Chapter 2], it follows from computational complexity theory that the problem of counting the number of solutions to an ILP is #P-complete [30, Section 7.3]. Nevertheless, as shown below, there are practical ILPs and MIPs for which the number of optimal solutions is > 1 but is not exceedingly large. We tested MORSE on such instances.

The problem of finding all optimal solutions to an ILP has been studied since the seminal paper of [31]. They restricted attention to $0-1$ ILPs, also known as *selection problems*, in which all variables are constrained to the values 0 and 1, representing decisions to select or exclude each variable. Examples include facility location and minimum hitting set. Their key insight was that each time a new optimum is found, one can add a constraint, known as a *cut*, to the ILP, so that the newly found optimal solution is no longer feasible in the adjusted problem, but all other optimal solutions remain both feasible and optimal. This yielded an effective, sequential method to enumerate optimal solutions, well-suited to the computers available in 1972. This method was later generalized to ILPs with integer-valued variables by [32]. Our approach perturbs the objective function, rather than modifying the constraints, and is better suited to parallel computers that are now available.

To our knowledge, the first paper to propose randomization for the problem of finding multiple optima was by [33]. Few subsequent papers have investigated randomization in this context; one such study is that of [6], in which a small experiment found a randomized method inferior to deterministic methods. The infrequent use of randomization for ILPs contrasts with linear programming, for which randomization has been essential to sensitivity analysis [34].

Sensitivity analysis, also called post-optimality analysis, of mathematical programs concerns the extent to which the coefficients or right-hand sides of constraints can be varied while retaining optimality. Here, we are instead concerned with finding tiny perturbations that break ties between optimal solutions without introducing new optima, though some informal connections between our work and past work on sensitivity analysis are worth noting.

The topic of sensitivity analysis for MIPs was reviewed by [35] and more recently by [36]. One general style of sensitivity analysis for MIPs that evaluates small perturbations is the *tolerance approach*, pioneered by Wendell [37] and in many related papers by him and his colleagues. In the tolerance approach, one question that has been studied by [36,38–40] is the extent to which one can vary one or more coefficients in an MIP or ILP with a unique optimal solution while preserving the optimality of an

 doi:10.20944/preprints202501.1518.v1

optimal solution to the unperturbed problem. The resulting interval (for one coefficient) or region (for more than one coefficient) is called the *tolerance region*. One general negative result is that for 0/1 MIPs it is NP-hard to determine the tolerance region of a single coefficient in the objective function [38].

There also exist some positive results on varying one or two coefficients. For example, [36] formulated the problem of finding the tolerance interval/region for one or two coefficients as an MIP itself. In the special case where the instance is a knapsack ILP, [40] provided a polynomial-time algorithm to find the tolerance interval for each individual coefficient.

1.3. Use of MIPLIB for Testing

To help gauge and stimulate progress in the ILP field, experts, beginning in 1992, have curated *MIPLIB*, a library of publicly available MIP problems. There efforts were inspired by *NETLIB*, a similar library for linear programs. The technical report on MIPLIB version 3.0 [41] explains the motivation to develop MIPLIB. Most recently updated in 2017, the current MIPLIB 6.0 set contains 1,065 problems [42]. All MIPLIB instances are assigned alphanumeric names, which may be descriptive (such as , a protein folding problem from the MIPLIB 2010 set [43]) or not descriptive (such as , described as "Geographic radar station allocation" in the MIPLIB 2017 set [42]). The aforementioned studies of [3–6] used MIPLIB to test their respective methods.

1.4. Our Contributions

In Section 2, we present the theory of our perturbation method and the evaluation metrics we use to test it. We implemented our method on top of the Gurobi MIP solver, and compared it to possible usages of the Gurobi `solution pool` method, the SCIP `count` method, and the CPLEX `populate` method. We tested MORSE largely on 20 published MIP instances. A sample experiment compares how many distinct optimal solutions are observed after r independent Gurobi runs, both with existing methods and with our method.

We evaluate the diversity of the solutions produced by MORSE, though the algorithm does not explicitly maximize the diversity as was done by [5,6]. In Section 3, we show that our perturbation method outperforms Gurobi's `solution pool` on MIPLIB and MadHitter instances, as quantified by the diversity metrics defined in Section 2 and by the number of distinct optima found. We conclude in Section 4 with a summary of our empirical results, some limitations, and some possible future directions.

2. Materials and Methods

2.1. Perturbing the Coefficients in the Objective Function

We present an algorithm, MORSE, to map random perturbations to the coefficients on the binary and integer objective function variables to efficiently find distinct optima for ILP instances with multiple optima. Some theory below assumes that variables are restricted to be integers, though our implementation allows continuous variables, which requires an extra check for optimality We refer to the Gurobi single solution or solution pool method as the conventional *homogeneous/uniform weights method*, in which we solve the problem in its original, unmodified form. In particular, Gurobi does not alter the original coefficients of binary or integer variables in the objective function. In practice, one can obtain multiple solutions via the homogeneous weights method by varying the Gurobi random seed and/or by using the solution pool. In our experiments, we varied the random seed and obtained one solution per Gurobi run to facilitate a head-to-head comparison with MORSE. We use the term *homogeneous/uniform weights* because leaving the original coefficients unchanged is equivalent to multiplying all the coefficients by 1 homogeneously or uniformly.

Central to MORSE is the selection of a small value $\varepsilon > 0$ that determines the maximum extent to which the coefficients may be varied, as explained first with an example and subsequently with formulae.

 doi:10.20944/preprints202501.1518.v1

Consider the following instance:

$$\begin{aligned} \text{maximize} \quad & x_1 + x_2 \\ \text{subject to the constraints} \quad & x_1, x_2 \leq 2, \\ & x_1 + x_2 \leq 3.5, \\ & x_1, x_2 \in \mathbb{Z}. \end{aligned}$$

The objective function can be expressed as $c^T x$ where $c = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$.

Suppose we choose $\varepsilon = 0.01$.

Next, we generate a random *perturbation vector* $v \overset{iid}{\sim} U(1-\varepsilon, 1+\varepsilon) = U(0.99, 1.01)$. If all variables are integral, then $\dim(v) = \dim(x)$, which is the number of variables in the objective function. If there are continuous variables, $\dim(v)$ is the number of binary or integer variables (defined formally below).

An example of such a perturbation vector is $v_1 = \begin{pmatrix} 0.99864 \\ 1.00142 \end{pmatrix}$.

We subsequently define the perturbed coefficient vector $c'_1 = v_1^T c = \begin{pmatrix} 0.99864 \\ 1.00142 \end{pmatrix}$.

This yields the perturbed objective function $c'^T_1 x = 0.99864x_1 + 1.00142x_2$. Without modifying the constraints, we solve the resulting instance and obtain the optimum $\{x_1 = 1, x_2 = 2\}$.

On another execution of MORSE, with the same value for ε, imagine that we generate the perturbation vector $v_2 = \begin{pmatrix} 1.00045 \\ 0.99312 \end{pmatrix}$. Following the same steps as above, we obtain the optimum $\{x_1 = 2, x_2 = 1\}$.

By executing two independent runs of MORSE, we can find the two distinct optima for the provided instance. To generalize this notion, for any instance with $s > 1$ distinct optima, we can try to find all distinct optima in $r \geq s$ independent MORSE runs.

2.2. Choosing ε

Let P be a MIP with objective function $\sum_{i=1}^{n} c_i x_i$ such that $c_i \neq 0$ and $x_i \in \mathbb{Z}$ for at least one index i. We assume that each variable has type $\texttt{vtype}(x)$, which is one of three possible types: *Binary*, *Integer* or *Continuous*. In MIPLIB instances, each variable x has specified lower and upper bounds that we denote by $\texttt{lb}(x), \texttt{ub}(x)$ respectively. The MadHitter instances we used for testing are ILP selection problems in which every variable in the objective function is of type *Binary* with lower bound 0 and upper bound 1.

Define $S = \sum_{i=1}^{n} f(c_i, x_i)$ where

$$f(c, x) = \begin{cases} |c| \cdot \max(|\texttt{lb}(x)|, |\texttt{ub}(x)|), & \texttt{vtype}(x) \neq \textit{Continuous}, \\ 0, & \texttt{vtype}(x) = \textit{Continuous}, \end{cases}$$

Then $\varepsilon = \frac{1}{25}$.

For example, suppose the objective function is $2x + 3y + z$, all variables are of type *Integer* and the variables are constrained to be in the following intervals $x \in [-2, 1], y \in [1, 4], z \in [-4, 2]$. Then, $S = (2 \times 2) + (3 \times 4) + (1 \times 4) = 20$, and $\varepsilon = 1/40 = 0.025$. If we scale the coefficients up by a factor of 5 to make the objective function $10x + 15y + 5z$, then $S = 100$ and $\varepsilon = 1/200 = 0.005$, decreasing by a factor of 5. If we scale the coefficients down by a factor of 5 to make the objective function $0.4x + 0.6y + 0.2z$, then $S = 4$ and $\varepsilon = 1/8 = 0.125$, increasing by a factor of 5. In general, scaling the objective function coefficients by a factor of $k > 0$ scales ε by a factor of $1/k$.

An advantage of this method is that it can be executed by simply examining the structure of the instance. If variable bounds are not available, then one can solve the instance to obtain a single optimum and use the absolute values of the variables in that solution to define ε, which can then be used to search for other optima.

 doi:10.20944/preprints202501.1518.v1

2.3. Applying the Perturbation, Pseudocode for MORSE

Define perturbation vector $v_1, \ldots, v_n \overset{iid}{\sim} U(1-\varepsilon, 1+\varepsilon)$. Define P', the perturbed instance of P, as an MIP with identical constraints as P (as defined in Section 2.2) and objective function

$$\sum_{i=1}^{n} c'_i x_i = \sum_{i=1}^{n} g(c_i, v_i, x_i)$$

where

$$g(c, v, x) = \begin{cases} cvx, & \texttt{vtype}(x) \neq \textit{Continuous}, \\ cx, & \texttt{vtype}(x) = \textit{Continuous}. \end{cases}$$

The full pseudocode for MORSE is shown below with Algorithms 1 and 2 as helper functions used in Algorithm 3.

Algorithm 1: Coefficient Scaling

Input : MILP
Output: equivalent MILP with integral coefficients

Function `ScaleCoefficient`($\mathcal{P}$: *MILP*):
    // Extract coefficients from the input MILP.
    $c_1 = \frac{a_1}{b_1}, \ldots, c_n = \frac{a_n}{b_n} \leftarrow$ `objective_coefficients`($\mathcal{P}$);
    // Compute Scaler from denominators of the coefficients.
    $s \leftarrow LCM(b_1, \ldots, b_n)$;
    // Create a copy of the input MILP.
    $\mathcal{Q} \leftarrow \mathcal{P}$;
    // Scale coefficient with the scaler.
    `objective_coefficients`($\mathcal{Q}$) $\leftarrow s \cdot$ `objective_coefficients`($\mathcal{P}$);
    return $\mathcal{Q}$;

Algorithm 2: Compute ε

Input : MILP
Output: ε the perturbation factor

Function `ComputeEpsilon`($\mathcal{P}$: *MILP*):
    $\mathcal{S} \leftarrow 0$;
    // Compute upper bound of the optimum w.r.t. non-continuous variables.
    foreach (c_i, x_i) in `objective`($\mathcal{P}$) **do**
        if `vtype`(x_i) in (`Integer`, `Boolean`) **then**
            $\mathcal{S} \leftarrow \mathcal{S} + |c_i| \cdot \max(|\texttt{lb}(x_i)|, |\texttt{ub}(x_i)|)$;
        end
    end
    return $\varepsilon \leftarrow \frac{1}{2\mathcal{S}}$

 doi:10.20944/preprints202501.1518.v1

Algorithm 3: MORSE Procedure

Input : MILP unoptimized, a random seed
Output: MILP optimized

Function `MORSE`$(\mathcal{P}, \phi)$**:**
  initialize random seed with ϕ;
  $\mathcal{Q} \leftarrow$ `ScaleCoefficient`$(\mathcal{P})$;
  $\varepsilon \leftarrow$ `ComputeEpsilon`$(\mathcal{Q})$;
  foreach (c_i, x_i) in `objective`$(\mathcal{Q})$ **do**
    if `vtype`(x_i) in (`Integer`, `Boolean`) **then**
      $v_i \sim Uniform(1-\varepsilon, 1+\varepsilon)$;
      $c_i' \leftarrow v_i \cdot c_i$;
    else
      $c_i' \leftarrow c_i$;
    end
  end
  `objective`$(\mathcal{Q}) = \{(c_i', x_i)\}$;
  `// Use MILP solver (e.g., Gurobi) to find the optimal solution to the conditioned MILP.`
  return $\leftarrow$ `MILPSolver`$(\mathcal{Q})$;

2.4. Some Theory to Justify the Choice of ε

Our intuition is that by making ε sufficiently small, any optimum with the perturbed objective function will also be an optimum for the original objective function. We prove that optima are preserved for ILPs with integer coefficients in Theorem 1. However, the converse is untrue; for any specific perturbation of the objective function, only some optima for the original problem may remain optima for the perturbed problem, as shown in the example above in which a perturbation preserves one possible optimum, but not the other. Our intuition is that some partial converse holds such that, for every optimum of the unperturbed problem, there exist perturbations that retain that optimum; this intuition is partly formalized in Theorems 2 and 3. When there are continuous variables, however, our implementation checks that optima are conserved, because this cannot be guaranteed.

Theorem 1. *Suppose all the coefficients in the original problem are integers, all the variables are binary or integer, and the problem P is a maximization problem. There exists a sufficiently small $\varepsilon > 0$ such that for any perturbation $c_1, \ldots, c_n \rightarrow c'_1, \ldots, c'_n$ and any resulting optimum $x'_1, \ldots, x'_n$ of the perturbed instance P', $x'_1, \ldots, x'_n$ is also an optimum of the original instance P.*

Proof. Let $x_1, \ldots, x_n$ be an optimum of P. Define $\varepsilon < \left(2\sum_{i=1}^{n} |c_i x_i|\right)^{-1}$ and let P' be the perturbed instance of P w.r.t. ε. Consider $x'_1, \ldots, x'_n$ such that $x'_1, \ldots, x'_n$ is optimal for P' but is not optimal for the original instance P. Due to the integrality constraints, the assumption that the coefficients are integers, the assumption that the variables are not continuous, and the assumption that the problem is a maximization problem,

$$P\left(x_1, \ldots, x_n\right) - P\left(x'_1, \ldots, x'_n\right) \geq 1. \tag{1}$$

 doi:10.20944/preprints202501.1518.v1

Consider what happens to these values in the perturbed problem $P^{'}$. Seeking a contradiction, suppose that x' has a better objective function value than x for P', then

$$\begin{aligned} P'\left(x^{'}{}_1,\ldots,x^{'}{}_n\right) - P'\left(x_1,\ldots,x_n\right) &\leq \left[P\left(x^{'}{}_1,\ldots,x^{'}{}_n\right) - P\left(x_1,\ldots,x_n\right)\right] \\ &\quad + \left[\left|P'\left(x_1,\ldots,x_n\right) - P\left(x_1,\ldots,x_n\right)\right|\right] \\ &\quad + \left[\left|P'\left(x^{'}{}_1,\ldots,x^{'}{}_n\right) - P\left(x'_1,\ldots,x'_n\right)\right|\right] \\ &< -1 + \frac{1}{2} + \frac{1}{2} \\ &< 0 \end{aligned} \tag{2}$$

Because we assumed the original problem P as a maximization problem, this is a contradiction. The first term in square brackets is at most -1 based on Equation (1). The second and third terms in square brackets are $< 1/2$ due to the definition of ε. □

In our experiments, most instances are minimization problems, due to the structure of MIPLIB, so we must also consider the complementary case in which P is a maximalization problem. For the latter case, the first numbered inequality (Equation (1)) becomes instead

$$P\left(x^{'}{}_1,\ldots,x^{'}{}_n\right) - P\left(x_1,\ldots,x_n\right) \geq 1$$

and the subsequent contradiction (Equation (2)) is that

$$P'\left(x_1,\ldots,x_n\right) - P'\left(x^{'}{}_1,\ldots,x^{'}{}_n\right) < 0$$

meaning that the original solution has a lower (better for minimization) objective function value than the alternative solution.

The assumption that all the coefficients for the integer-constrained variables are themselves integral does not hold for some MIPLIB instances. The idea behind the proof of Theorem 1 can be generalized to fractional coefficients by scaling up the problem by multiplying all the coefficients by the least common multiple of the denominators. For example, if the objective function is $1.3x + 3.2y + 2.1z$, this is equivalent to $\frac{13}{10}x + \frac{16}{5}y + \frac{21}{10}z$, and the least common multiple of the denominators is 10. We can scale up the objective function to $13x + 32y + 21z$ to make the adjusted coefficients integral and then, Theorem 1 applies to the adjusted problem. As shown in the examples above in 2.2, the consequence of multiplying the coefficients by 10 is to divide ε by 10. The scaling is described at lines 4–10 of the pseudocode. However, scaling does not work and the Theorem does not apply if there are irrational coefficients.

For the proof of Theorem 1, we assumed that one optimal solution is available and we used the bound $\varepsilon < \left(2\sum_{i=1}^{n}|c_i x_i|\right)^{-1}$ without assuming any bounds on the variables x_i. If upper and lower bounds on variables are available, as is the case for MIPLIB, then we can instead use the setting in Section 2.2 and the same reasoning applies. Using the available bounds on the variables simplifies MORSE because one does not need a sequential phase of finding one optimal solution before the subsequent phase of running multiple parallelized runs to obtain multiple optimal solutions.

Next, we prove, in theory, that MORSE can find any optimal solution.

Theorem 2. *For any optimum $x_1,\ldots,x_n$ of P, there exists a perturbation $c'_1,\ldots,c'_n$ such that $x_1,\ldots,x_n$ is also an optimum of P'*

 doi:10.20944/preprints202501.1518.v1

Proof. Choose the identity perturbation such that $c'_1 = c_1, \ldots, c'_n = c_n$. Then $P = P'$ and optima are conserved. □

Next, we consider the effects of non-identity perturbations in breaking ties.

Theorem 3. *Suppose the instance P is defined as in Theorem 1. For any optimum $x_1, \ldots, x_n$ of P, there exists a perturbation $c'_1, \ldots, c'_n$ such that $c'_1 \neq c_1, \ldots, c'_n \neq c_n$ and $x_1, \ldots, x_n$ is also an optimum of P'.*

Proof. Let ε be defined as in the proof of Theorem 1. Define $c'_i = \begin{cases} (1+\varepsilon) \cdot c_i, & c_i x_i > 0, \\ (1-\varepsilon) \cdot c_i, & c_i x_i < 0. \end{cases}$

If the values of x_i remain unchanged, then the constraints of P' remain satisfied. Moreover, any solution that satisfies the constraints of P also satisfies the constraints of P'. By this perturbation, $c'_1 x_1 + \ldots + c'_n x_n > c_1 x_1 + \ldots + c_n x_n$. Hence, because of the choice of ε, the absolute net change in the objective function value, which is given by $\varepsilon \sum_{i=1}^{n} |c_i x_i|$, is less than 1.

Aiming to find a contradiction, suppose that some other solution $\hat{x}_1, \ldots, \hat{x}_n$ yields a higher objective function than $x_1, \ldots, x_n$. Consider the difference in objective function value of this solution for the original instance P and the perturbed instance P'. For each of the k terms involving an integer-constrained variable, if $c_i \hat{x}_i > 0$, then $|c_i \hat{x}_i - c'_i \hat{x}_i| \leq \varepsilon c_i \hat{x}_i$. Therefore, the net increase in objective function value for $\hat{x}_1, \ldots, \hat{x}_n \leq \varepsilon \sum_{i=1}^{n} |c_i x_i|$. Thus, the solution $\hat{x}_1, \ldots, \hat{x}_n$ cannot have a strictly better objective function value in the permuted problem P' than does the original solution $x_1, \ldots, x_n$. □

From Theorem 3 it follows that the probability of finding any particular optimal solution is > 0. Nevertheless, the result is weak because the volume around the optimal solution in the proof depends on ϵ and hence inversely on the coefficients. Next, we prove in two steps a stronger result about MORSE's capability to find different optima for the special case of 0–1 selection problems.

Lemma 1. *Let P be a 0/1 ILP that is a selection, minimization problem with objective function $\sum_{i=1}^{n} c_i x_i$, and every coefficient $c_i = 1$. Suppose one perturbs the objective function by multiplying each coefficient by some $p_i \sim U(1-\varepsilon, 1+\varepsilon)$ as the* MORSE *method does to arrive at the new problem $\sum_{i=1}^{n} p_i x_i$. Then any optimal solutions $\{x_{j_1}, x_{j_2}, \ldots, x_{j_d}\}$ to the perturbed problem are optimal solutions to the original problem. The indices of the selected variables have the additional property that the sum of perturbed coefficients $\sum_{k=1,d} p_{j_k}$ corresponding to the variables that are selected is minimized among all possible sums of d perturbed coefficients.*

Proof. Since P is a selection problem, there must be some positive integer value d, such that all optimal solutions are of size d. By Theorem 1, any optimal solution to the perturbed problem is a solution to the original selection problem. Suppose the variables that are set to 1 and hence selected in an optimal solution to the original problem are $\{x_{j_1}, x_{j_2}, \ldots, x_{j_d}\}$. Then, the objective function value of that solution in the original problem is d and the objective function value of that solution in the perturbed problem is $\sum_{k=1}^{d} p_{j_d}$. Any solution that assigns 1 to exactly d variables in the perturbed problem will have an objective function value that is the sum of the perturbed coefficients for the d selected variables. Therefore, the optimal solutions of the perturbed problem are precisely those that minimize $\sum_{k=1}^{d} p_{j_d}$. It is possible that multiple solutions to the perturbed problem are tied for the minimum sum, although this is unlikely in practice. □

Theorem 4. *Let P be a 0/1 ILP that is a selection, minimization problem with objective function $\sum_{i=1}^{n} c_i x_i$, and every coefficient $c_i = 1$. Suppose one perturbs the objective function by multiplying each coefficient by*

 doi:10.20944/preprints202501.1518.v1

some $p_i \sim U(1-\varepsilon, 1+\varepsilon)$ as the MORSE method does. Suppose (as explained in the Proof of Lemma 1) that all optimal solutions have objective function value d meaning that exactly d variables have value 1. Suppose $\{x_{j_1}, x_{j_2}, \ldots, x_{j_d}\}$ and $\{x_{k_1}, x_{k_2}, \ldots, x_{k_d}\}$ are two distinct optimal solutions of the original problem. Then, the probabilities that $\{x_{j_1}, x_{j_2}, \ldots, x_{j_d}\}$ and $\{x_{k_1}, x_{k_2}, \ldots, x_{k_d}\}$ are optimal solutions of one problem perturbation from MORSE are equal.

Proof. Let M_j be the set of perturbations that make $V_j = \{x_{j_1}, x_{j_2}, \ldots, x_{j_d}\}$ optimal for the perturbed problem. Let M_k be the set of perturbations that make $V_k = \{x_{k_1}, x_{k_2}, \ldots, x_{k_d}\}$ optimal for the perturbed problem. We show constructively that there is a 1-to-1 correspondence $C : M_j \to M_k$. It follows that the probability that a uniformly randomly selected perturbation is in M_j equals the probability that a uniformly randomly selected perturbation is in M_k. Let p_j be a perturbation of coefficients in M_j. We construct a corresponding perturabtion p_k in M_k. Specifically, the function C maps p_j to p_k by permuting indices without changing the set of uniformly selected multipliers. For example, suppose there are three variables and p_j perturbs the coefficients by multiplying c_1 by 0.9998, c_2 by 1.0007, and c_3 by 0.9993. Then, an example of a permuted perturbation would be instead to multiply c_1 by 1.0007, to multiply c_2 by 0.9993 and c_3 by 0.9998. For $i = 1, \ldots n$, we have two cases:

(1) If x_i is the rth variable x_{jr} in V_j then, C maps the multiplier of $c_i = c_{jr}$ to be instead the multiplier of c_{kr}.
(2) If x_i is the sth variable absent from V_j then, C maps the multiplier of c_i to be instead the multiplier for the coefficient of the sth variable absent from V_k, preserving the value of the perturbed coefficient.

Since the mapping C preserves the values of the coefficients and of the objective function in the perturbed problem, it follows from Lemma 1 that C preserves optima of the perturbed problem. Since C preserves the values of the variables and any permutation of the dimensions is a 1-to-1 correspondence on the set of dimensions, it follows that C is a 1-to-1 correspondence between M_j and M_k. □

We are also interested in the probability that each possible optimum of the original problem appears as an optimum of a perturbed problem. Suppose there are v variables that appear in the objective function, each corresponding to a distinct term. It thus follows that the space of perturbations is isomorphic to the hypercube $\mathcal{Q} := [1-\varepsilon, 1+\varepsilon]^v$. The hypercube representation is reminiscent of the work of [31] on multiple optima for binary programs, though there exist a few notable differences. Their work focused exclusively on the vertices of the hypercube, each of which represented a different solution, 0's for the non-selected variables and 1's for the selected variables. By contrast, our interest extends to both the boundary and interior of $\mathcal{Q}$, where $\mathcal{Q}$ instead represents perturbations to the original objective function, rather than solutions. Employing the hypercube formalism, one may ask:

For any perturbation, which solutions remain optimal?

Under what conditions on the perturbation $p \in \mathcal{Q}$ is the solution to the perturbed problem for p unique?

For any optimum x to the original, unperturbed problem, does there exist a perturbation p, such that if one applies p, then x remains an optimum to the perturbed problem? [We gave an affirmative answer under some assumptions in Theorem 3; it would be desirable to determine if these assumptions can be weakened.]

(a) If so, we can define $V(x)$ as the volume within the hypercube of perturbations that preserve the optimality of x. Can one determine upper and lower bounds on $V(x)$?

Our intuition, which we tested empirically via the `solution pool` method, suggests that for the majority of perturbations, the solution to the perturbed problem is unique for the variables constrained to be integers. However, there must be boundary regions of zero volume at which two or more

 doi:10.20944/preprints202501.1518.v1

solutions are tied. In Theorem 4, we answered **Question 3(a)** affirmatively for selection problems. We hypothesize that for some subset of the problems we consider, the answer to **Question 3** is positive, and this is verifiable for MIPLIB instances for which the number of distinct optima is known. If all optima of the unperturbed problem are identified, then one can use repeated sampling of perturbations to empirically estimate the probability that each optimal solution is found.

2.5. Conceptual Comparison Between MORSE and a Previously Tested Perturbation Method

Our perturbation method differs from a similar method originally proposed by [34] in the context of linear programming and later implemented for binary/integer problems by [6]. In the latter work, each binary and integer coefficient in the objective function is multiplied by the quantity $(1 + \varepsilon\xi_{ij})$, where ε is a positive constant and $\xi_{ij} \overset{iid}{\sim} U(-1, 1)$. In both cited studies, the choice of ε did not depend on the input instance. As a consequence, optima may not be preserved; indeed, the fact that small perturbations can change optimal solutions was a key point on non-robustness that Ben-Tal and Nemirovski made in their perturbation analysis. Trapp and Konrad [6] tested their version of a perturbation method on just three instances, and only for *fixed* values of ε, and found that the results were not favorable for finding all optimal solutions. Trapp and Konrad did not prove that their perturbation preserved optimal solutions, although their intent was to preserve optima or near-optima. In contrast, our MORSE method multiplies the coefficient for each binary and integer variable by a random real number in the range $[1 - \varepsilon, 1 + \varepsilon]$, where ε is determined *adaptively* based on the size of the coefficients in the objective function and the bounds on the objective function variables (see Section 2.2). As shown above, the perturbed instances generated by MORSE preserve optimal solutions.

2.6. MIPLIB as a Source of Instances for Testing

We downloaded select instances from MIPLIB to test our weighted perturbation method on a diverse set of optimization problems. We chose instances from the following overlapping lists:

1. Instances mentioned by Danna and colleagues in their 2007 study [4]
2. Instances mentioned by Danna and Woodruff [5]
3. Instances mentioned by Trapp and Konrad [6]
4. Instances that include binary and/or integer decision variables and classified as "easy" on MIPLIB (solvable in one hour or less by a commercial solver on default settings). Instances formally retired prior to the MIPLIB 2017 set (for which the task of finding one optimum had become "too easy") were still considered for our testing, as the task of solving for multiple/all optima remains interesting and sufficiently complex. Such instances were obtained from the MIPLIB 1996 [41], 2003 [44], or 2010 [43] sets.

To be considered for our central testing stage, instances needed to meet the following additional criteria:

5. The objective function includes binary and/or integer variables.
6. The instance has multiple optima.

To enforce Criterion 5, algorithmic testing was needed, and was performed as follows:

a. Iterate through the variable types for all variables in the objective function.
1. If at least one variable type is binary or integer, accept the instance. Otherwise, discard the instance.

The pseudocode is shown in Algorithm 4.

 doi:10.20944/preprints202501.1518.v1

Algorithm 4: Check Objective Function

Input : MILP
Output: Acceptance/rejection decision

```
Function CheckObjFunction(P: MILP):
    foreach x_i in objective(P) do
        if vtype(x_i) in (Integer, Boolean) then
            // Accept the instance.
            return True
        end
    end
    // Reject the instance
    return False
```

Criterion 6 could be could be partially evaluated by referencing the number of optima listed for each instance in [4]. Instances pulled from this paper were confirmed to have multiple optima as explained in Section 3. Additional testing was needed to confirm the existence of multiple optima for instances from [5] or [6] that did not intersect the list of instances in [4]. If our script detected that an instance did not satisfy any of the above criteria, the instance was dropped from further consideration.

2.7. Figures of Merit Used in Testing

As explained in Section 1, there are three freely-available methods against which we can compare: the `solution pool` in Gurobi, the `count` function in SCIP, and the `populate` function in CPLEX. Since we implemented MORSE using Gurobi, most of our tests were executed in comparison to the `solution pool` method.

We compared the performance of our perturbation method against the standard, unperturbed method using the following test design. We executed r runs in parallel and obtained s optimal solutions, not necessarily distinct. We considered two figures of merit.

First, we examined the number of distinct optima found at least once among the s optimal solutions. Two solutions are considered distinct if and only if they differ in the value of at least one variable appearing in the objective function.

Second, we compared the solution sets on two previously-used measures of diversity: *Hamming distance* and *Shannon entropy*. Solution set diversity has been previously quantified by average Hamming distance between solution vectors (see [5,6]).

Definition 1 (Hamming Distance). *For solution vectors $x = [x_1, \ldots, x_n]$ and $y = [y_1, \ldots, y_n]$, the Hamming distance $H(x, y)$ is given by:*

$$H(x,y) = \sum_{i=1}^{n} f(x_i, y_i) \quad \textit{where } f(x,y) = \begin{cases} 1, & x \neq y \\ 0, & x = y \end{cases}$$

For example, consider solution vectors $A = [1, 2, 3, 4]$ and $B = [1, 0, 3, 5]$; $H(A, B) = 2$. The Hamming distance does not consider the magnitude of the difference between integer variables, and is increased by a uniform increment of 1 for each integer variable that differs between solutions.

Definition 2 (Shannon entropy). *Let $\mathcal{X} = \{x_1, x_2, \ldots, x_{k \leq t}\}$ be the set of solution vectors, the Shannon entropy is defined as follows.*

$$S(\mathcal{X}) = -\sum_{x \in \mathcal{X}} p_x \log_2(p_x)$$

 doi:10.20944/preprints202501.1518.v1

In some other applications, Shannon entropy is denoted by H, but we use the mnemonic S for Shannon entropy and reserve the mnemonic H for Hamming distance. For example, if there exist two solutions that are found with probabilities $1/3$ and $2/3$, then the Shannon entropy is as follows.

$$S(\mathcal{X}) = -\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) \approx 0.918$$

In our analysis, we are interested in the entire solutions as combinatorial objects. If instead there are three solutions and each solution occurs with probability $1/3$, then the Shannon entropy of the solution set is:

$$S(\mathcal{X}) = -3\left(\frac{1}{3}\log_2\frac{1}{3}\right) \approx 1.585$$

Suppose, instead, we sample 6 not necessarily distinct solutions, and we observe the distinct solutions A, B, C 3 times, 2 times, and 1 time respectively. Then, the Shannon entropy of the solution multiset would be:

$$-\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{3}\log_2\frac{1}{3} + \frac{1}{6}\log_2\frac{1}{6}\right) \approx 1.459$$

We found one analysis of a problem-specific method to find multiple optima that used (higher) Shannon entropy on the values of individual variables among solutions as a figure of merit [10]. However, we did not find any previous study of multiple optima in MIPs that used Shannon entropy on the multiplicity of solutions, as we have done. In general, higher Hamming distance and higher Shannon entropy are preferable in that they indicate higher diversity. When comparing the Hamming distance or Shannon entropy for solution sets found by the two methods, we do not normalize by the number of distinct solutions found.

3. Results

We verified the correctness of the MORSE method and tested its performance empirically on the 20 MIPLIB instances summarized in Table 1. For the first set of large-scale tests, in parallel, we executed 100 runs with MORSE and 100 Gurobi runs with homogeneous (unchanged) weights and collected one optimal solution for each run. Runs were sorted using (as the sort key) either the random weights seed (for MORSE runs) or the Gurobi random generator seed (for homogeneous weights runs). From two side-by-side orderings of homogeneous weights runs and MORSE runs, we aggregated lists of optima by taking one optimum from each run. Each list may contain duplicates because the runs were executed in parallel, in contrast to the inherently sequential approach of [31,32] described in Section 1.

We further ordered the paired lists of optima in three distinct ways:

1. *Seed-based*: Keeping the random number generator seed-based ordering outlined above.
2. *Random shuffle*: Randomly shuffling the optima collected using the above *Seed-based* order.
3. *Greedy for diversity*: Randomly choosing a solution from the Seed-based list to be the first solution reported. Then, for each next solution chosen, choosing the solution that maximizes the average pairwise Hamming distance with all previously selected solutions.

Our first test measured diversity as quantified by the Hamming distances for binary and integer variables between pairs of optima selected. For each instance, we built a plot of average pairwise Hamming distance normalized by the total binary solution size (y-axis) vs. solution number (x-axis) for both MORSE and uniform weights for each of the three methods described above. For every sorting method, the average pairwise Hamming distance between optima found with MORSE exceeded the pairwise Hamming distance of optima found with the uniform weights method at almost all run numbers (see Figures 1–5 below). The shaded gray area surrounding the lines gives the 95% confidence interval.

 doi:10.20944/preprints202501.1518.v1

Table 1. List of tested MIPLIB instances, the highest-numbered version of MIPLIB in which each instance appears, and the corresponding number of distinct optima. MIPLIB versions 2.0 and 3.0 were released in 1996, version 4.0 was released in 2003, version 5.0 was released in 2010, and version 6.0 was released in 2017. Distinct optima were enumerated using the PySCIPOpt `count` function [3]. Instances marked with * are also found in Table 1 of [4], and the number of optima claimed in that table is consistent with the number of optima found by `count`. Distinct optima counts marked with $\geq$ give the number of optima enumerated by `count` within a 24-hour runtime; if there is no $\geq$, then enumeration was fully completed in runtime $<$ 24 hours.

MIPLIB instance	Most recent MIPLIB version	Number of distinct optima
`set1al`	2.0	2
`set1cl`	2.0	2
`p0201`	3.0	*4
`bell3a`	3.0	5
`mod008`	3.0	*6
`p0033`	3.0	*9
`lp41`	2.0	24
`vpm2`	4.0	33
`khb05250`	3.0	51
`stein9`	2.0	54
`stein45`	3.0	*70
`supportcase14`	6.0	256
`supportcase16`	6.0	256
`stein15`	3.0	315
`stein27`	3.0	*2106
`harp2`	5.0	$\geq$ 1
`pigeon-10`	6.0	$\geq$ 6574
`app2-2`	6.0	$\geq$ 41819
`noswot`	6.0	$\geq$ 48776
`vpm1`	3.0	$\geq$ 86186

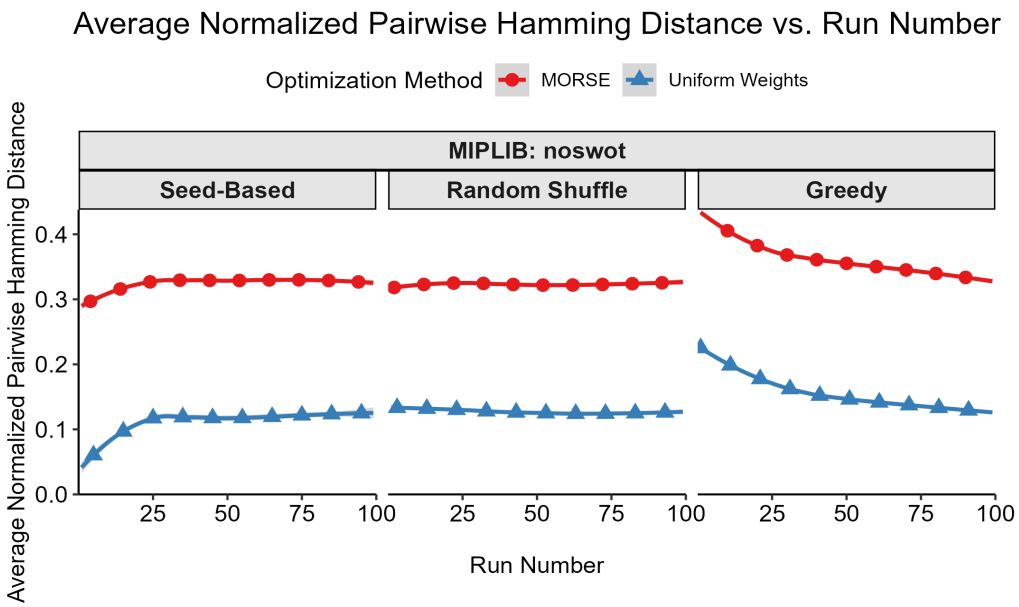

Figure 1. Average Normalized Pairwise Hamming Distance (y-axis) vs. Run Number (x-axis) for MIPLIB instance `noswot`.

 doi:10.20944/preprints202501.1518.v1

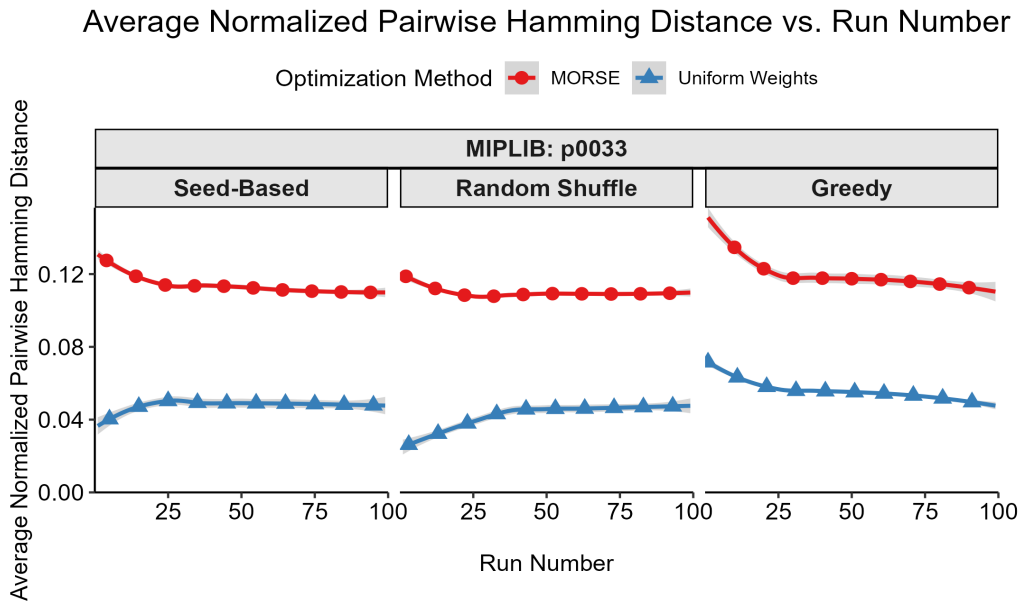

Figure 2. Average Normalized Pairwise Hamming Distance (y-axis) vs. Run Number (x-axis) for MIPLIB instance `p0033`.

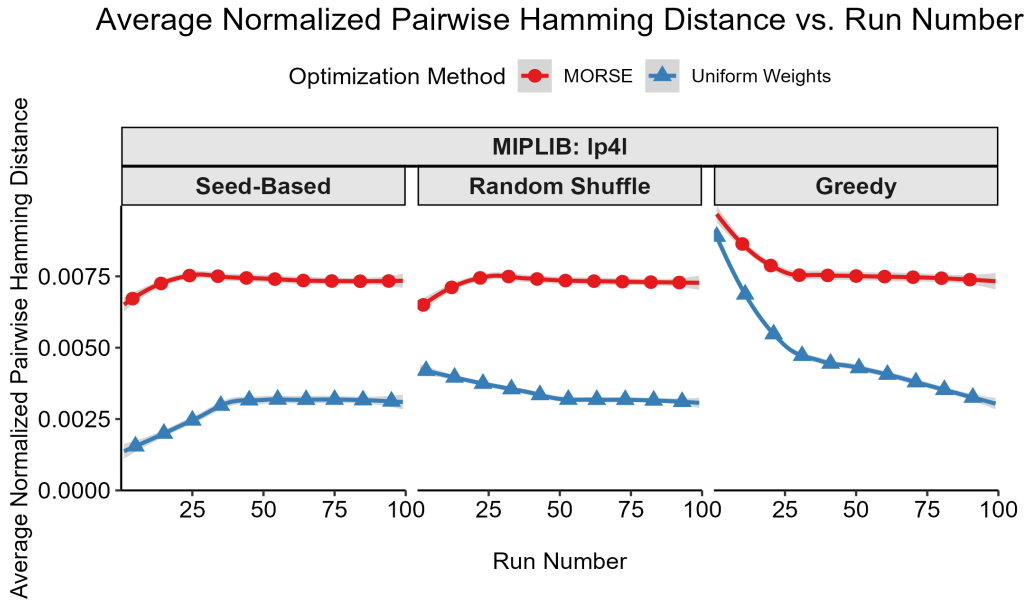

Figure 3. Average Normalized Pairwise Hamming Distance (y-axis) vs. Run Number (x-axis) for MIPLIB instance `lp4l`.

 doi:10.20944/preprints202501.1518.v1

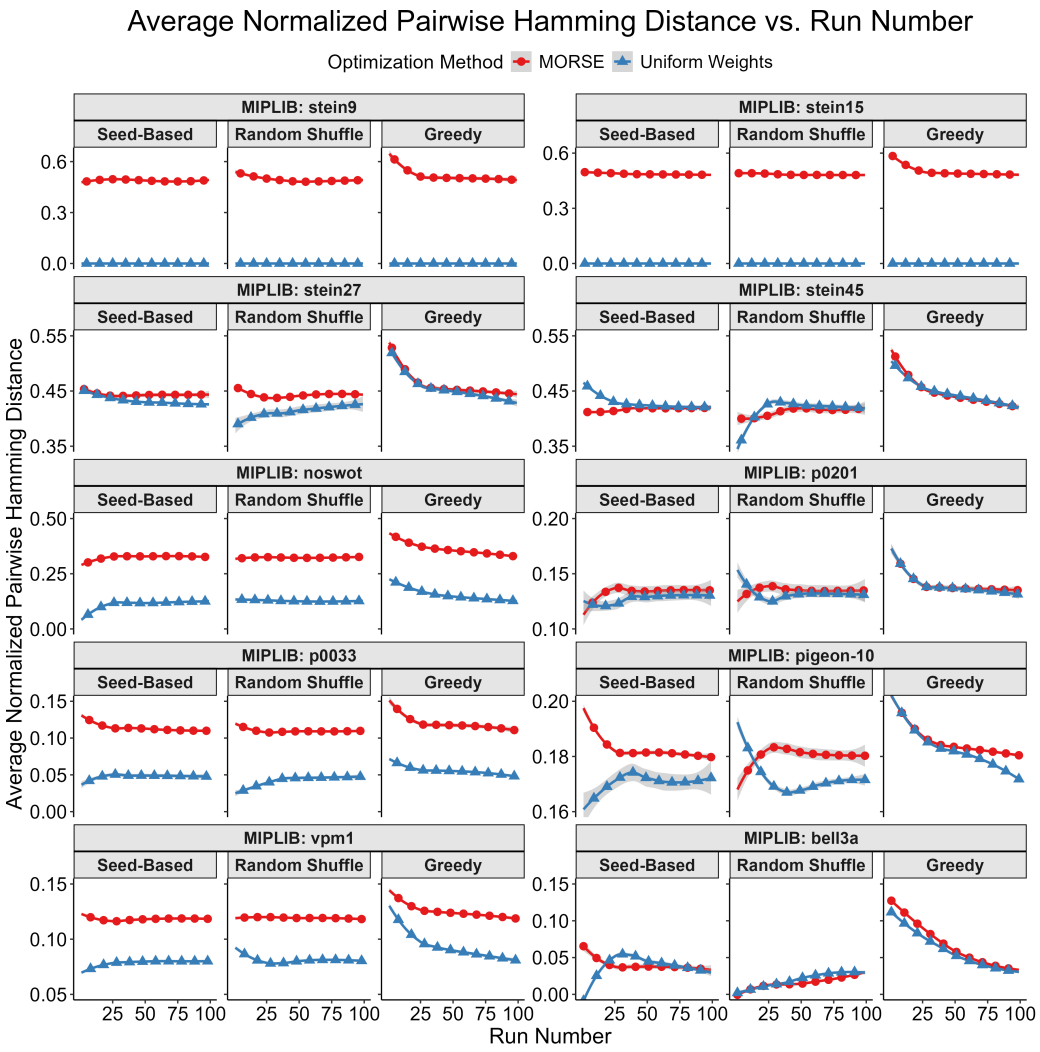

Figure 4. Tabular results. Average Normalized Pairwise Hamming Distance (y-axis) vs. Run Number (x-axis) for tested MIPLIB instances with the 10 highest maximum average normalized pairwise Hamming distances. The plots are arranged in descending order of maximum average normalized pairwise Hamming distance. Please note that the y-axis starts above 0 for some panels.

 doi:10.20944/preprints202501.1518.v1

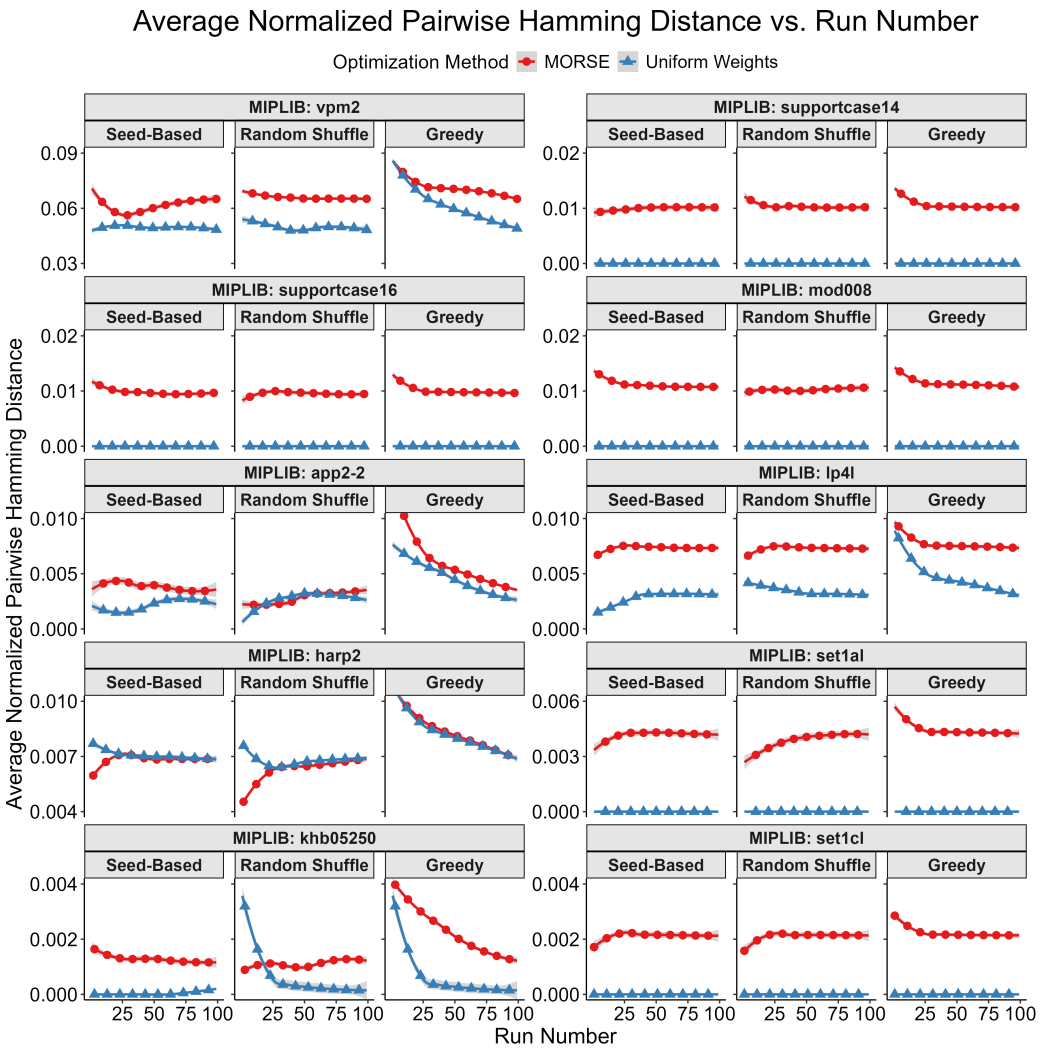

Figure 5. Tabular results, Average Normalized Pairwise Hamming Distance (y-axis) vs. Run Number (x-axis) for tested MIPLIB instances with the 10 lowest maximum average normalized pairwise Hamming distances. The plots are arranged in descending order of maximum average normalized pairwise Hamming distance. Please note that the y-axis starts above 0 for some panels.

The second test measured the number of distinct optima found as a function of run number. We formally define "distinct" optimal solutions as those that differ on variables in the objective function; two solutions with identical objective function variable values, but different non-objective function variable values, are considered identical for this test. The motivation for focusing on the objective function variables is that they are the variables of interest in applications such as MadHitter. If each optimum has an equal probability of being found (see bf Question 3 above), then the expected number of runs s needed to find all optima grows as $slog(s)$ [45, Section 2.4, pp. 32–33]. For the second test, we executed r runs with MORSE and r runs with uniform weights and collected 1 optimum for each run. The value of r ranged from 100 to 2000, depending on the number of distinct optima for a given instance.

We then plotted the number of distinct optima (y-axis) found vs. run number (x-axis). This test was best suited for instances with < 100 distinct optima, all of which we were able to find in a reasonable number of runs, thus allowing us to compare the performance of the two weights methods in finding distinct optima. Each test result fit one of the two following scenarios:

1. While we cannot find all optima in the allotted r runs, we observe that we find more optima with MORSE than with the uniform weights method. (see Figures 6 and 7)
2. We find the same number of optima in the allotted r runs with both methods, but fewer runs are needed with MORSE than with the uniform weights method (see Figure 8)

 doi:10.20944/preprints202501.1518.v1

Both numbered scenarios favor MORSE, so we conclude empirically that our method performs better than the homogeneous weights method on the second test.

We tested the method of [4] as implemented in the CPLEX `populate` function on the 15 instances in Table 1 for which the exact number of distinct optima is known. We initially set the `PopulateLim` parameter, which controls the maximum number of solutions returned for the python interface, to the known number of optima for each instance. With that setting, CPLEX found all optima for 12/15 instances, but only found 1 out of 2 known solutions for the instances `set1al` and `set1c1`, as well as 1 out of 51 solutions for the instance `khb05250`. With PopulateLim set to higher values, CPLEX found the second solutions for `set1al` and `set1cl` but did not find more than 1 solution for `khb05250` (Table 2). We conclude that this method works well on most instances if one knows the number of solutions beforehand (as is rarely the case in most practical settings), but does not work perfectly.

Table 2. Tested MIPLIB instances for which the CPLEX `populate` function did not find all optimal solutions.

MIPLIB instance	PopulateLim setting	Number of distinct optima (CPLEX)	Number of distinct optima (SCIP)	Number of missed optima
khb05250	51	1	51	50
khb05250	102	1	51	50
khb05250	204	1	51	50
khb05250	408	1	51	50
khb05250	816	1	51	50
khb05250	1632	1	51	50
khb05250	3264	1	51	50
khb05250	6528	1	51	50
khb05250	13056	1	51	50
khb05250	26112	1	51	50
set1al	2	1	2	1
set1cl	2	1	2	1

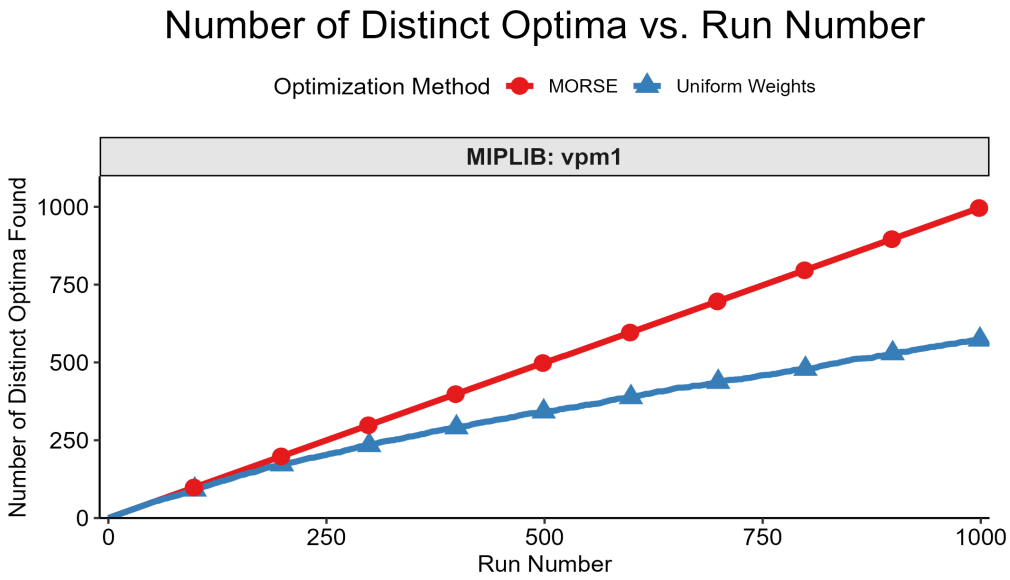

Figure 6. Number of Distinct Optima vs. Run Number for MIPLIB instance `vpm1`. While we cannot find all optima in the allotted 1,000 runs, we find 998 distinct optima with MORSE and 573 distinct optima with the uniform weights method.

 doi:10.20944/preprints202501.1518.v1

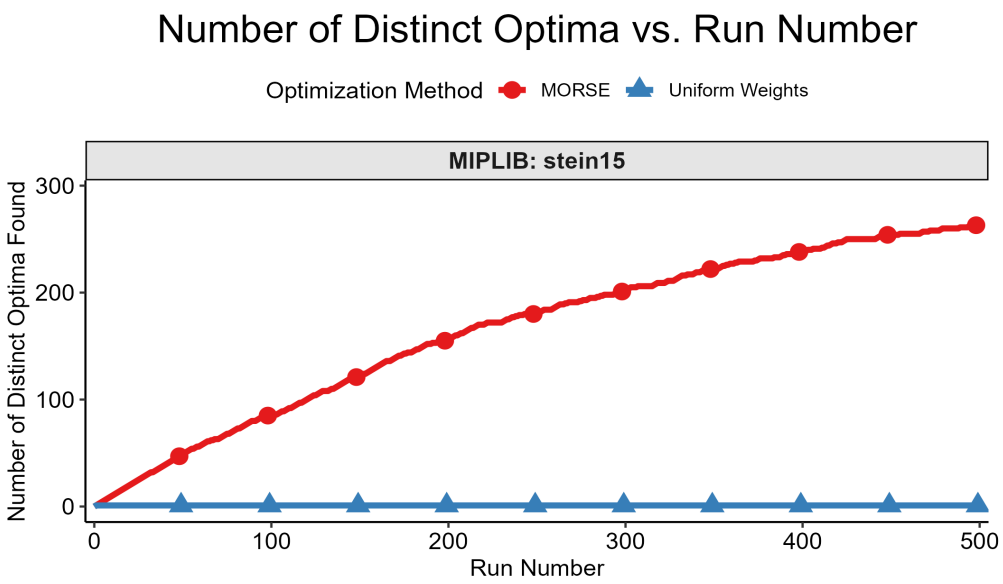

Figure 7. Number of Distinct Optima vs. Run Number for MIPLIB instance `stein15`. With MORSE, we find 264 distinct optima in the allotted 500 runs, while we find one optimum with the uniform weights method.

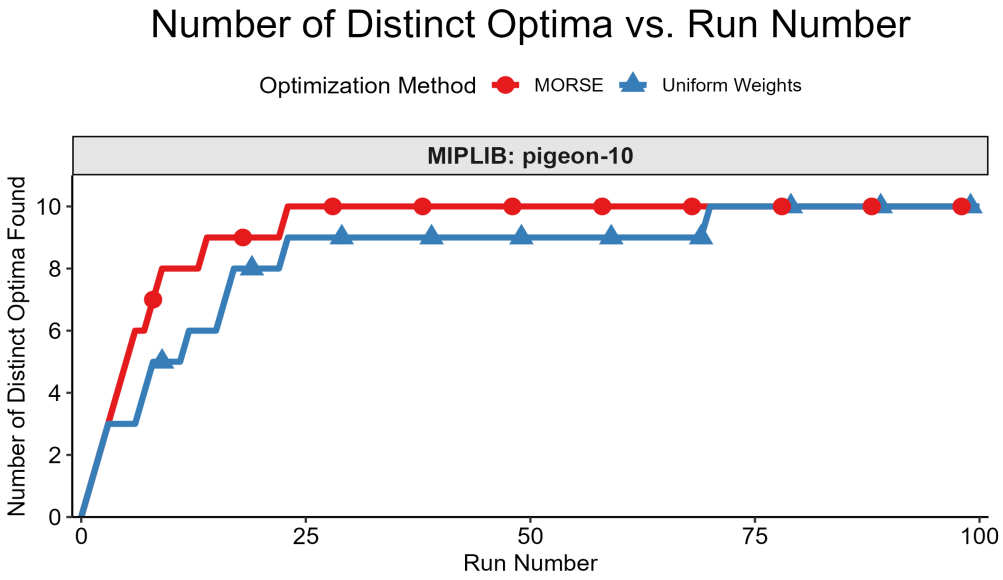

Figure 8. Number of Distinct Optima vs. Run Number for MIPLIB instance `pigeon-10`. With MORSE, we find 10 distinct optima in 23 runs, while 70 runs are needed to find 10 distinct optima using the uniform weights method.

 doi:10.20944/preprints202501.1518.v1

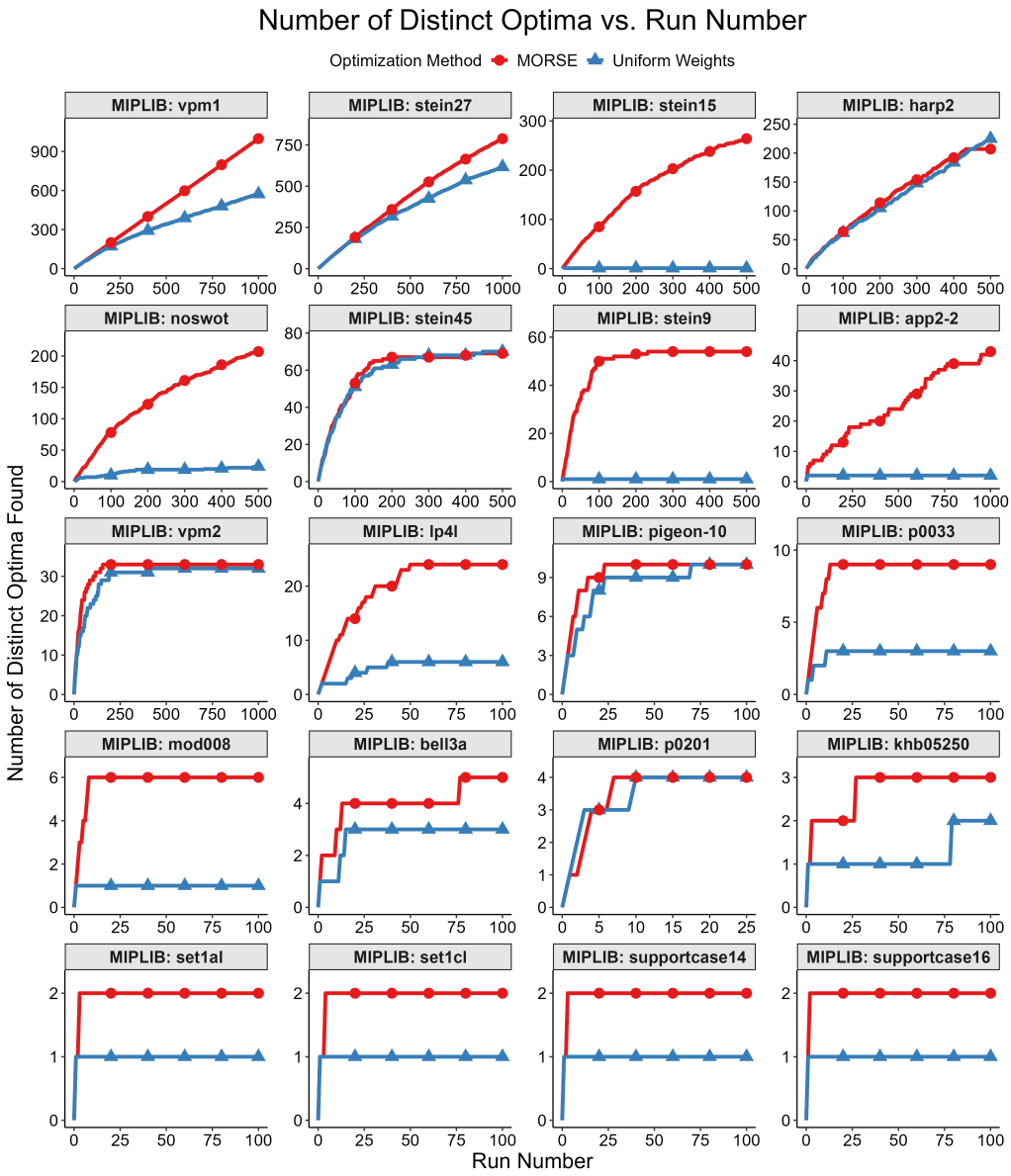

Figure 9. Full MIPLIB results: Number of Distinct Optima vs. Run Number for all MIPLIB instances tested. The plots are arranged in descending order of maximum number of unique optima found.

In addition to testing MORSE on MIPLIB instances, we tested on instances from MadHitter, an algorithm that identifies gene targets for experimental CAR-T, conjugated antibody, and coated nanoparticle therapies [1]. The algorithm takes single-cell tumor and non-tumor data for a set of patients and returns the smallest set of target genes that kills at least a specified *lower bound* percentage of each patient's tumor cells (for example, $\geq$ 80%) while killing at most a specified *upper bound* percentage of each patient's non-tumor cells (for example, $\leq$ 10%). The single-cell datasets were reduced to focus on the 1,269 genes encoding cell surface receptors, which may serve as targets for one or multiple of the experimental therapies listed above [1]. The objective is to minimize the number of treatments that a patient must receive. In a second layer of optimization for a cohort of multiple patients, we optimize the total number of distinct gene targets used, such that the number of targets for each patient is optimal. Intuitively, if a single patient has multiple sets of gene targets of the same optimal size, we select the optimal set that best permits us to reuse the same targets for other patients in the cohort; the exact ILP formulation is given in [1].

For each feasible instance (dataset, lower bound, and upper bound permutation for which an optimum exists), we executed 1,000 runs for both MORSE (in which each gene is randomly assigned a

 doi:10.20944/preprints202501.1518.v1

weight in the range $[1-\varepsilon, 1+\varepsilon]$) and the uniform weights method (in which each gene has a uniform weight of 1). Since all solutions found were composed of fewer than 100 targets, $\varepsilon = 0.01$ was used for all instances. This value of ε was chosen such that the maximum magnitude of the sum of perturbations, given by $\varepsilon \cdot$(*number of targets in solution*), did not exceed 1.

Just as we did for the MIPLIB instances, we measured the number of distinct optima found as a function of run number. Distinct solutions were defined as solutions that differed globally; we did not account for differences in local (patient) variables. For example, consider *global optimum A*, given by [G1, G2, G3], with corresponding local optima as follows: *patient 1* receives the treatments [G1, G2] and *patient 2* receives the treatments [G2, G3]. Next, consider global optimum B, also given by [G1, G2, G3], with corresponding local optima as follows: *patient 1* receives the treatments [G1, G3], and *patient 2* receives the treatments [G1, G2]. Global optimum A and global optimum B are considered identical for the purposes of this test. Results with many choices for the number of runs on the x-axis and two individual datasets can be found in Figures 10 and 11. Results with fewer choices for the number of runs for nine datasets can be found in Figure 12.

A practical motivation for looking at multiple optima of MadHitter instances is to identify which genes occur in any optimal solution or in all optima. An observed example of the impact of MORSE is conveyed by analyzing the brain cancer dataset `GSE103322`. With default MadHitter parameters and using the solution listing delineated above, the gene *CELSR1* (short for cadherin EGF LAG seven-pass G-type receptor 1) appears (by chance) in multiple optima generated by MORSE, including the second optimum. However, when using Gurobi's solution pool, *CELSR1* appears for the first time in optimum number 397. The gene *CELSR1* has previously been shown to have higher messenger RNA levels in brain cancer as compared to surrounding non-cancerous brain tissue by [46,47], so this is a a favorable gene to select as a possible target in an optimal treatment.

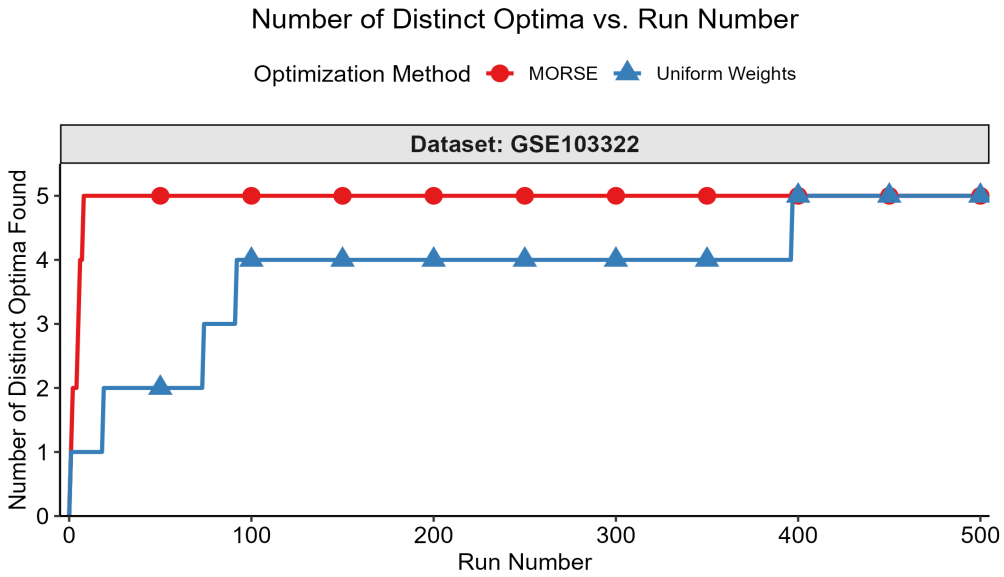

Figure 10. Number of Distinct Optima Found (y-axis) vs. Run Number (x-axis) for `GSE103322` (brain cancer) 1269-gene dataset, lower bound = 0.8, upper bound = 0.1. With MORSE, we find all 5 distinct optima in 9 runs. With uniform weights, we find these optima in 397 runs.

 doi:10.20944/preprints202501.1518.v1

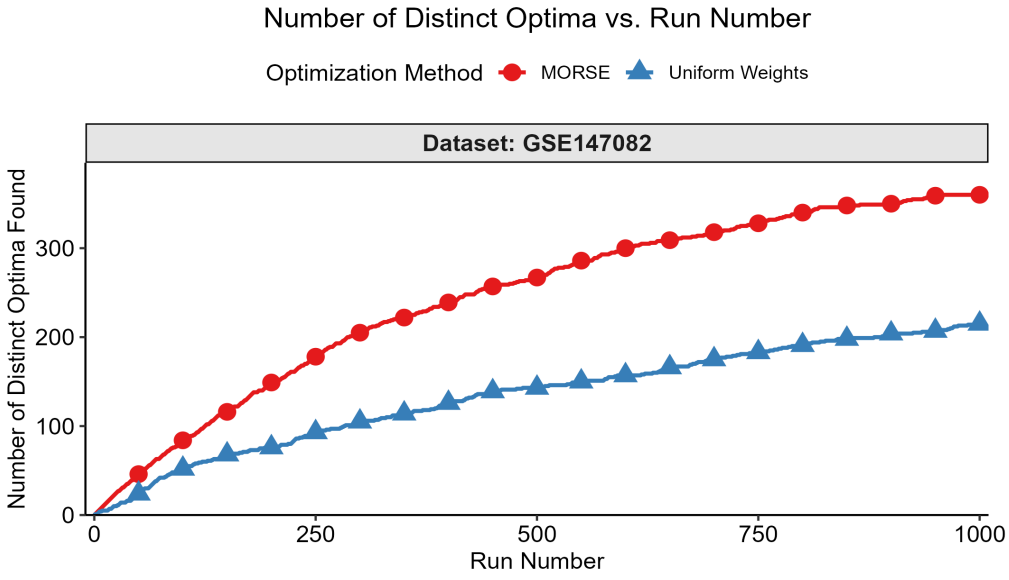

Figure 11. Number of Distinct Optima Found (y-axis) vs. Run Number (x-axis) for GSE147082 (ovarian cancer) 1269-gene dataset, lower bound = 0.8, upper bound = 0.1. With MORSE, we find all 360 distinct optima in 959 runs. With uniform weights, we find 215 optima in 1,000 runs.

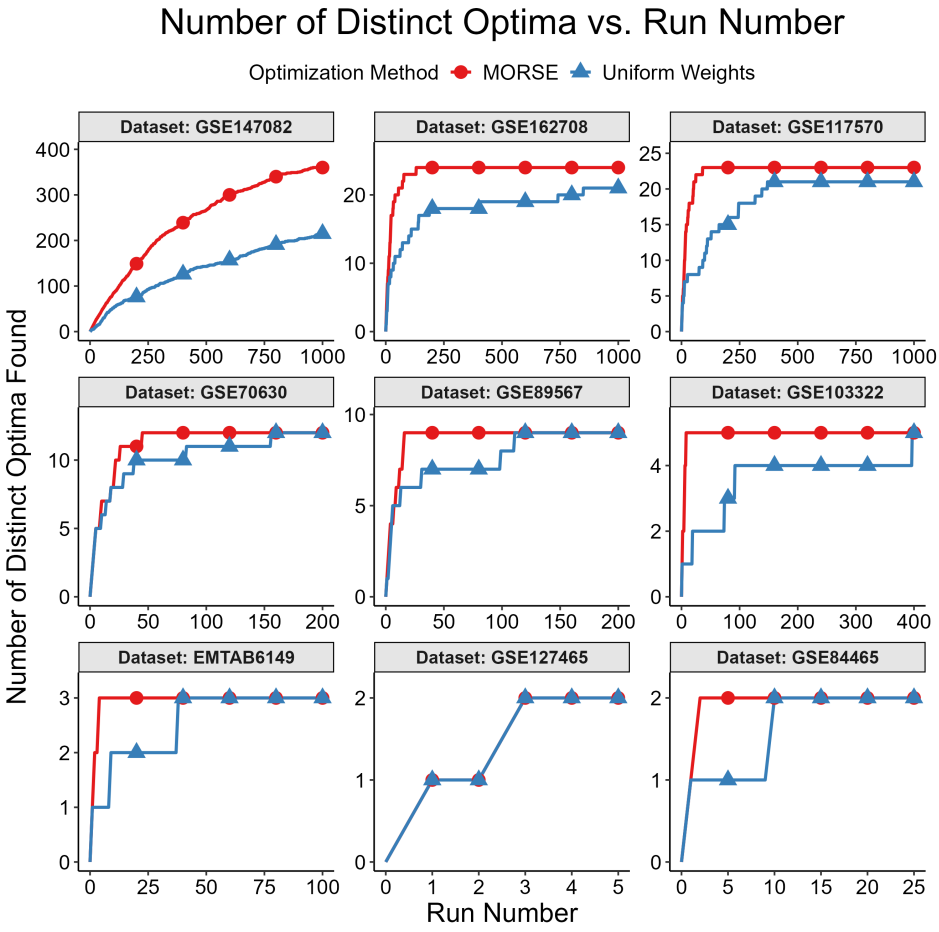

Figure 12. Tabular results. Number of Distinct Optima Found (y-axis) vs. Run Number (x-axis) for all 1269-gene cancer datasets tested, lower bound = 0.8, upper bound = 0.1. For GSE127465 (lower row, center), the results for MORSE (red) and uniform weights (blue) are identical

 doi:10.20944/preprints202501.1518.v1

To quantify bias in the solution space, we used Shannon entropy. A higher Shannon entropy value corresponds to a higher level of randomness and a lower level of bias in the distribution. In the present context, a high Shannon entropy value is desirable, as the algorithm should not favor any particular solution in the solution space. To find Shannon entropy for each feasible permutation, we assigned an empirical normalized probability p_i to each gene solution in the solution set and applied the standard Shannon entropy formula (see Section 2.7).

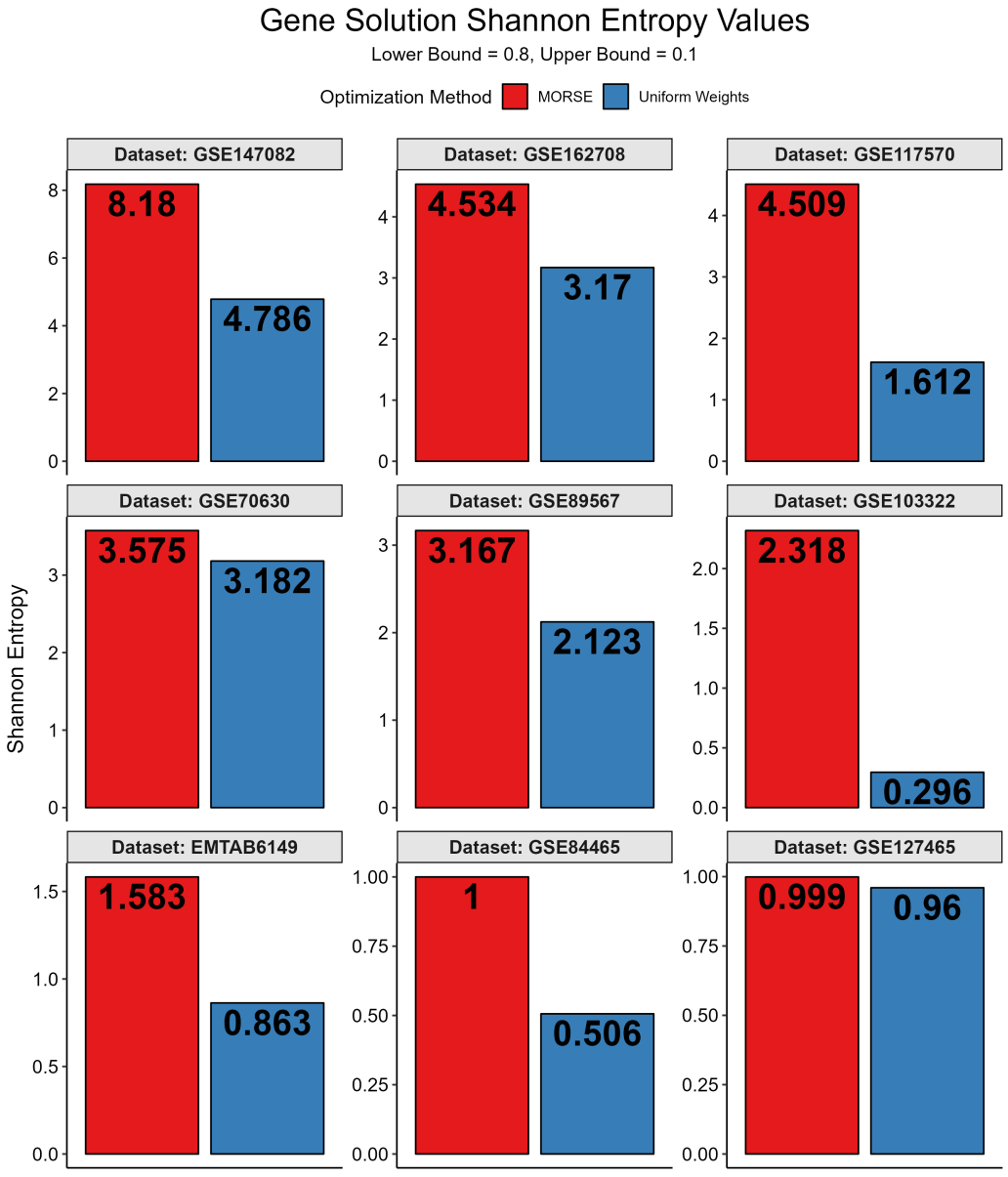

Figure 13. Gene Solution Shannon Entropy Values for datasets with feasible solutions at lower bound = 0.8, upper bound = 0.1. The plots are arranged in descending order of maximum Shannon entropy. Shannon entropy for MORSE runs exceeds that of uniform weights runs for all datasets used.

For the MIPLIB instances tested, we also employed Shannon entropy on the optima found with both MORSE and the uniform weights method. We calculated Shannon entropy on the multiplicity of each optimum found (Figure 14) by assigning an empirical normalized probability p_i to each distinct optimum found. Additionally, for each instance tested, we calculated Shannon entropy on each variable by assigning p_i to each distinct variable value, then averaged the Shannon entropy score for each variable to find the average Shannon entropy (Figure 15). Both tests only considered the variables in the objective function. We found that, for both approaches, the Shannon entropies for optima found

 doi:10.20944/preprints202501.1518.v1

with MORSE generally exceeded those of the optima found with uniform weights, except for one instance (harp2) where the latter marginally exceeded the former.

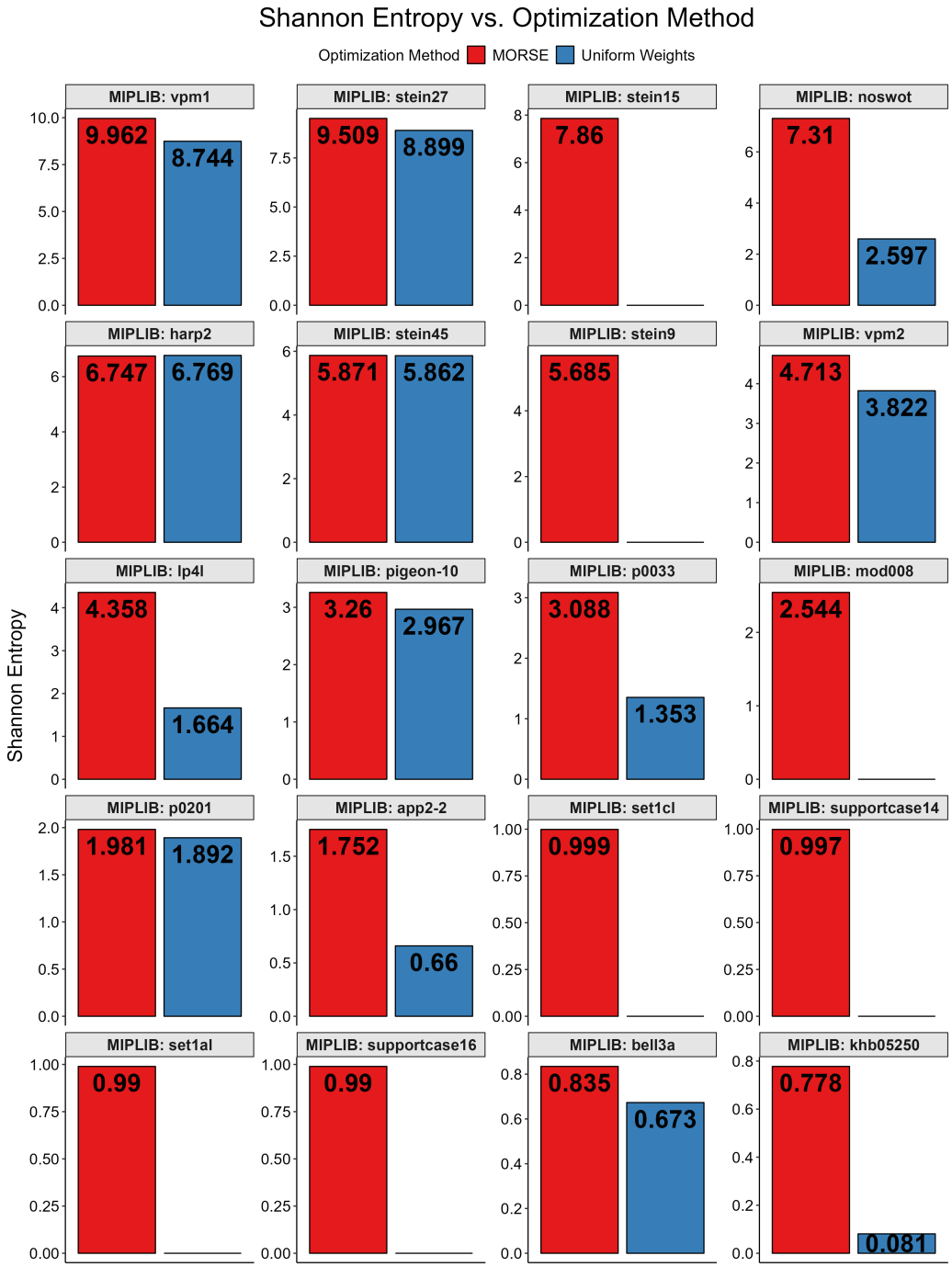

Figure 14. Full MIPLIB results, Shannon Entropy vs. Optimization Method for all MIPLIB instances tested. The plots are arranged in descending order of maximum Shannon entropy.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 21 January 2025 doi:10.20944/preprints202501.1518.v1

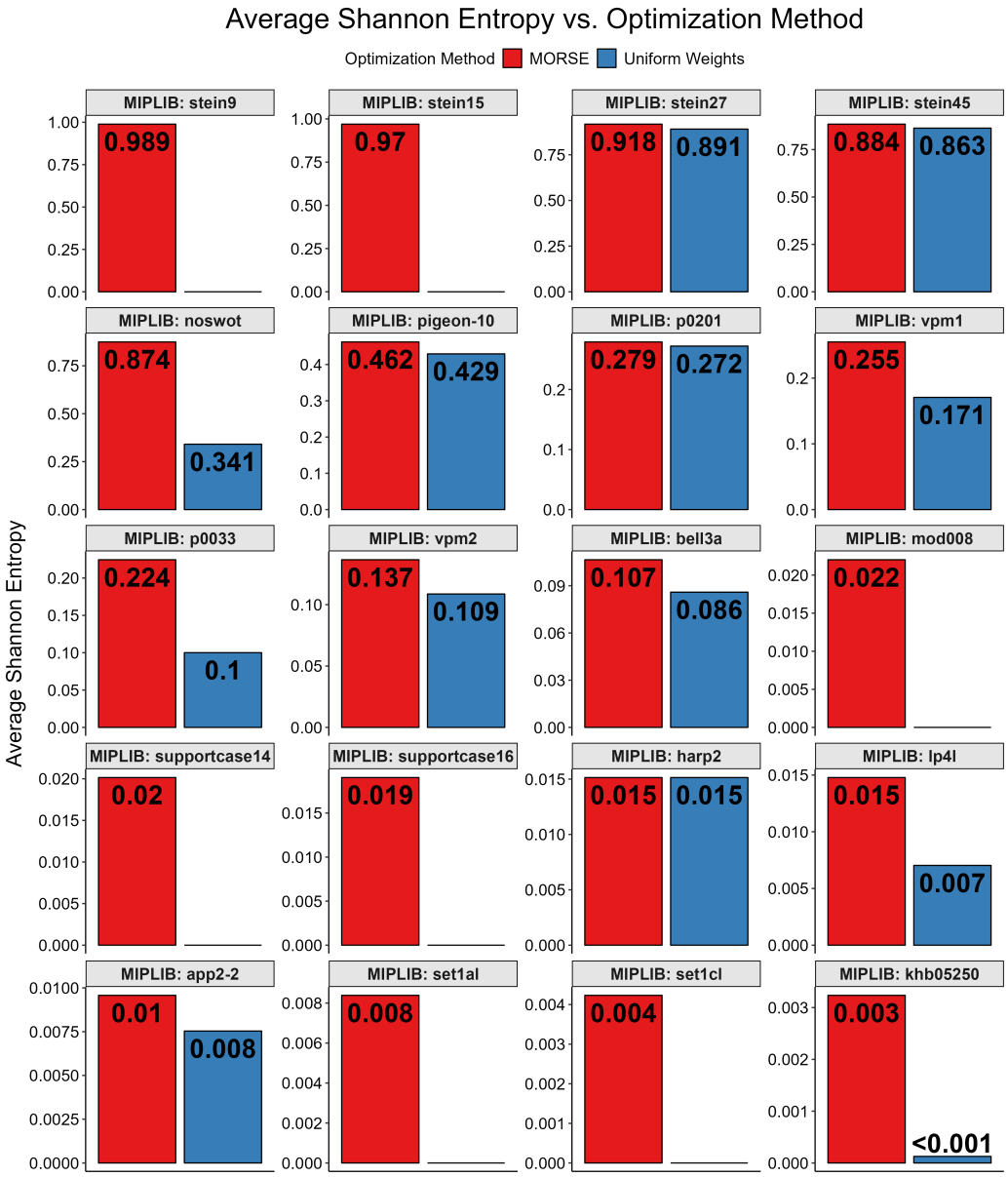

Figure 15. Full MIPLIB results, Average Shannon Entropy vs. Optimization Method for all MIPLIB instances tested. The plots are arranged in descending order of maximum average Shannon entropy.

4. Discussion

We implemented and evaluated MORSE, a perturbation method to find multiple optima for ILPs. After [33], few subsequent papers have investigated randomization in this context (see Section 1.2). A key advantage of the present work and other randomized methods is that they are amenable to parallelization. Another distinguishing feature is that we proved that our perturbation method preserves optima when all the coefficients are integers, which Trapp and Konrad did not prove for their implementation of a randomized algorithm ([6]).

With MORSE, we generally find more diverse optima (as quantified by higher average pairwise Hamming distance between solution vectors) within the first 100 distinct optima returned for MIPLIB instances with > 100 optima. Additionally, when executing n runs in parallel, we find distinct optima at a faster rate with MORSE than with the method in Gurobi. Based on the number of optima and the number of runs allotted for a given instance, we either find the same number of optima in fewer runs or find a greater number of optima using MORSE. An important practical application of our method is to decide quickly if an instance has more than one optimal solution. For example, in Figure 9, subpanels for `supportcase14` and `supportcase16`, we see two instances in which Gurobi's `solution`

 doi:10.20944/preprints202501.1518.v1

`pool` method suggests that the optimal solution is unique, but MORSE finds two distinct optimal solutions. Similarly, the CPLEX `populate` function returned just one optimum each for the MIPLIB instances `set1a1` and `set1c1`, despite setting the parameter values to return both optima. Moreover, the SCIP `count` function requires multiple steps to decide if there is more than one optimal solution: first solving to optimality, and then constructing and solving a modified ILP.

We also integrated our method with MadHitter, an algorithm that identifies minimal gene target hitting sets for experimental gene therapy cancer treatments by formulating the problem as an ILP. Similar to the result we obtained when testing on MIPLIB instances, we either found all optimal gene target sets, or a greater number of optimal gene target sets, with MORSE after executing $1,000$ runs in parallel compared to Gurobi. We quantified the Shannon entropy of each vector of $1,000$ runs for each dataset and weights method tested, and found that, for any given dataset, the Shannon entropy for MORSE runs exceeded that of uniform weights runs, indicating a higher diversity among the optima found by MORSE.

4.1. Limitations

The proposed method is not applicable to MIP problems in which all of the variables in the objective function are continuous, nor is it applicable to (Continuous) Linear Programming problems, in which all decision variables may take on continuous values. A key limitation is that he MORSE method does not guarantee to find all optima even for ILPs.

An additional limitation of the proposed method is that it is less amenable to computing environments in which parallel computing is not supported, or in which doing $p > 1$ runs in parallel costs substantially more than one run in whatever currency (e.g., money or computing allocation) is used to discourage users from executing many runs in parallel. In the SLURM-based computing environments (at the National Institutes of Health and Northwestern University) in which we conducted testing, submitting several runs in parallel is only mildly discouraged by a reduction in priority for jobs, so our highly parallelized method works well.

4.2. Future Directions

Given the desire of biologists and biochemists to find all optima to problems of interest, the present algorithm could be applied to domain-specific instances in the field of biology, such as haplotype inference [7] and other applications listed in Section 1. There also exists the possibility of developing other domain-specific sampling schemes, naturally leading to more extensive explorations of the permutation hypercube described in Section 2.4.

Other relevant recent studies, for example by [48–51], focus on finding a collection of k solutions that are approximately optimal and maximizing their diversity (measured by the sum of distances between solutions). The underlying optimization problems may be either in P or are NP-hard, and these studies connect the search for multiple solutions to other algorithmic concepts such as dynamic programming dispersion and fixed-parameter tractability. Their goal is not necessarily to find all optimal solutions, but instead to find a pre-specified number of high-quality, diverse solutions. Accordingly, explicitly optimizing diversity of the solutions produced could be an interesting future direction for our work as well, especially on instances with a very large collection of optimal or near-optimal solutions.

Author Contributions: "Conceptualization, P.S., S.K., and A.A.S.; methodology, N.S., P.S., E.R., S.K., and A.A.S.; software, N.S., P.S., and A.A.S.; validation, N.S.; formal analysis, N.S., P.S., A.A.S.; investigation, N.S., P.S., E.R., S.K., and A.A.S.; resources, N.S., E.R., S.K., and A.A.S.; data curation, N.S. and A.A.S; writing—original draft preparation, N.S., P.S., and A.A.S.; writing—review and editing, N.S., P.S., E.R., S.K., and A.A.S.; visualization, N.S.; supervision, E.R., S.K., and A.A.S.; project administration, S.K. and A.A.S.; funding acquisition, E.R. and S.K. All authors have read and agreed to the published version of the manuscript.", please turn to the CRediT taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

 doi:10.20944/preprints202501.1518.v1

Funding: This research was supported in part by the Intramural Research Program of the National Institutes of Health, National Cancer Institute. This work utilized the computational resources of the NIH Biowulf HPC cluster (http://hpc.nih.gov). Additionally, this research was supported in part through the computational resources and staff contributions provided for the Quest high performance computing facility at Northwestern University which is jointly supported by the Office of the Provost, the Office for Research, and Northwestern University Information Technology.

Data Availability Statement: The MORSE algorithmic code can be found at https://github.com/ruppinlab/MORSE. The input files, scripts, and testing data used to produce the results/figures presented in this manuscript can be found at https://github.com/ruppinlab/MORSE/tree/main/MORSE_data

Conflicts of Interest: The authors declare no conflicts of interest.The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ILP	integer linear program
MIP	mixed integer (linear) program
MORSE	Multiple Optima via Random Sampling and careful choice of the parameter Epsilon

References

1. Ahmadi, S.; Sukprasert, P.; Vegesna, R.; Sinha, S.; Schischlik, F.; Artzi, N.; Khuller, S.; Schäffer, A.A.; E., R. he landscape of receptor-mediated precision cancer combination therapy via a single-cell perspective. *Nat. Comm.* **2022**, *13*, 1613.
2. Gainer-Dewar, A.; Vera-Licona, P. The minimal hitting set generation problem: Algorithms and computation. *SIAM J. Discr. Math.* **2017**, *31*, 63–100.
3. Achterberg, T.; Heinz, S.; Koch, T. Counting solutions of integer programs using unrestricted subtree detection. In Proceedings of the Proceedings of 5th International Conference for Integration of AI and OR techniques in constraint programming for combinatorial optimization problems (CPAIOR), 2008, Vol. 5015, *Lecture Notes in Computer Science*, pp. 278–282.
4. Danna, E.; Fenelon, M.; Gu, Z.; Wunderling, R. Generating multiple solutions for mixed integer programming problems. In Proceedings of the Proceedings of International Conference on Integer Programming and Combinatorial Optimization (IPCO), 2007, Vol. 4513, *Lecture Notes in Computer Science*, pp. 280–294.
5. Danna, E.; Woodruff, D.L. How to select a small set of diverse solutions to mixed integer programming problems. *Oper. Res. Lett.* **2009**, *37*, 255–260.
6. Trapp, A.C.; Konrad, R.A. Finding diverse optima and near-optima to binary integer programs. *IIE Trans.* **2015**, *47*, 1300–1312.
7. Lippert, R.; Schwartz, R.; Lancia, G.; Istrail, S. Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Brief. Bioinform.* **2001**, *3*, 23–31.
8. Ballerstein, K.; von Kamp, A.; Klamt, S.; Haus, U. Minimal cut sets in a metabolic network are elementary modes in a dual network. *Bioinformatics* **2012**, *28*, 381–387.
9. Cai, J.; Liao, X.; Mao, Y.; Wang, R.; Li, H.; Ma, H. Designing gene manipulation schedules for high throughput parallel construction of objective strains. *Biotech. J.* **2023**, *18*, e2200578.
10. Robaina-Estévez, S.; Nikoloski, Z. On the effects of alternative optima in context-specific metabolic model predictions. *PLoS Comput. Biol.* **2017**, *13*, e1005568.
11. von Kamp, A.; Klamt, S. Enumeration of smallest intervention strategies in genome-scale metabolic networks. *PLoS Comput. Biol.* **2014**, *10*, e1003378.
12. Andersen, J.; Flamm, C.; Merkle, D.; Stadler, P.F. Chemical transformation motifs – modeling pathways as integer hypeflows. *IEEE Trans. Comput. Biol. Bioinform.* **2019**, *16*, 510–523.
13. Arthur, J.L.; Hachey, M.; Sahr, K.; Huso, M.; Kiester, A.R. Finding all optimal solutions to the reserve site selection problem: formulation and computational analysis. *Env. Ecol. Stat.* **1997**, *4*, 153–165.
14. Hädicke, O.; Klamt, S. Computing complex metabolic intervention strategies using constrained minimal cut sets. *Metab. Eng.* **2011**, *13*, 204–213.

15. Haus, U.; Klamt, S.; Stephen, T. Computing knock-out strategies in metabolic networks. *J. Comput. Biol.* **2008**, *15*, 259–268.
16. Hvidsten, T.R.; Lægreid, A.; Komorowski, J. Learning rule-based models of biological process from gene expression time profiles using gene ontology. *Bioinformatics* **2003**, *19*, 1116–1123.
17. Jensen, K.; Broeken, V.; Lærke-Hansen, A.; Sonnenschein, N.; Herrgård, N. OptCouple: Joint simulation of gene knockouts, insertions and medium modifications for prediction of growth-coupled strain design. *Metab. Eng. Comm.* **2019**, *8*, e00087.
18. Li, L.; Bansal, M. An integer linear programming solution for the domain-gene-species reconciliation problem. In Proceedings of the ACM BIBCB 2018, 2018, pp. 386–397.
19. Rodriguez-Miller, P.; Poupin, N.; de Blasio, C.; Le Cam, L.; Jourdan, F. DEXOM: Diversity-based enumeration of optimal context-specific metabolic networks. *PLoS Comput. Biol.* **2021**, *17*, e1008730.
20. Ruckerbauer, D.E.; Jungreuthmayer, C.; Zanghellini, J. Design of optimally constructed metabolic networks of minimal functionality. *PLoS ONE* **2014**, *9*, e92583.
21. Trinh, C.T.; Wlaschin, A.; Srienc, F. Elementary mode analysis: A useful metabolic pathway analysis tool for characterizing cellular metabolism. *Appl. Microbiol. Biotech.* **2009**, *81*, 813–826.
22. Vera-Licona, P.; Bonnet, E.; Brillot, E.; Zinovyev, A. OCSANA: Optimal combinations of interventions from network analysis. *Bioinformatics* **2013**, *29*, 1571–1573.
23. Voll, P.; Jennings, M.; Hennen, M.; Shah, N.; Bardow, A. The optimum is not enough: A near-optimal solution paradigm for energy systems synthesis. *Energy* **2015**, *82*, 446–456.
24. Wang, B.; Zhou, Y.; Zhou, Y.; Xu, S.; Niu, B. Diverse competitive design for topology optimization. *Struct. Multidisc. Opt.* **2018**, *57*, 891–902.
25. Lin, M.; Chang, P.; Lee, S.; Graeb, H. DeMixGen: Deterministic mixed-signal layout generation with separated analog and digital signal paths. *IEEE Trans. Comput.-Aided Des. Integr.. Circ. Syst.* **2016**, *35*, 1229–1242.
26. Oulasvirta, A.; Dayaman, N.R.; Shiripour, M.; John, M.; Karrenbauer, A. Combinatorial optimization of graphical user interface designs. *Proc. IEEE* **2020**, *108*, 434–464.
27. Church, R.L.; Baez, C.A. Generating optimal and near-optimal solutions to facility location problems. *Envir. Plan B: Urban Anal. City Sci.* **2020**, *47*, 1014–1030.
28. Arink, N.; Figueiredo, R.; Labatut, V. Efficient enumeration of the optimal solutions to the correlation clustering problems. *J. Glob. Optim.* **2023**, *86*, 355–391.
29. Schrijver, A. *Theory of Linear and Integer Programming*; John Wiley & Sons: New York, 1986.
30. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman: San Francisco, 1979.
31. Balas, E.; Jeroslow, R. Canonical cuts on the unit hypercube. *SIAM J. Appl. Math.* **1972**, *23*, 61–69.
32. Tsai, J.; Lin, M.; Hu, Y. Finding multiple solutions to general integer linear programs. *Eur. J. Oper. Res* **2008**, *184*, 802–809.
33. Galle, P. Branch & sample: A simple strategy for constraint satisfaction. *BIT Numer. Math.* **1989**, *29*, 395–408.
34. Ben-Tal, A.; Nemirovski, A. Robust solutions of linear programming problems contaminated with uncertain data. *Math. Prog,* **2000**, *88*, 411–424.
35. Sotskov, Y.N.; Leontev, V.K.; Gordeev, E.N. Some concepts of stability analysis in combinatorial optimization. *Discr. Appl. Math.* **1995**, *58*, 169–190.
36. Andersen, K.A.; Boomsma, T.K.; Nielsen, L.R. MILP sensitivity analysis of the objective function coefficients. *INFORMS J. Opt.* **2022**, *5*, 92–109.
37. Wendell, R.E. The tolerance approach to sensitivity analysis in linear programming. *Man. Sci.* **1985**, *31*, 564–578.
38. Van Hoesel, S.; Wagelmans, A. On the complexity of postoptimality analysis of 0/1 programs. *Discr. Appl. Math.* **1999**, *91*, 251–263.
39. Dawande, M.W.; Hooker, J.N. Inference-based sensitivity analysis for mixed interger/linear programming. *Oper. Res.* **2000**, *48*, 623–634.
40. Pisinger, D.; Saidi, A. Tolerance analysis for 0-1 knapsack problems. *Eur. J. Oper. Res.* **2017**, *258*, 866–876.
41. Bixby, R.; Ceria, S.; McZeal, C.M.; Savelsbergh, M.W.P. An updated mixed integer programming library: MIPLIB 3.0. Technical report, Georgia Institute of Technology, 1996.
42. Gleixner, A.; Hendel, G.; Gamrath, G.; Achterberg, T.; Bastubbe, M.; Berthold, T.; Christophel, P.; Jarck, K.; Koch, T.; Linderoth, J.; et al. MIPLIB 2017: Data-driven compilation of the 6th mixed-integer programming library. *Math. Progr. Comput.* **2021**, *13*, 443–490.

43. Koch, T.; Achterberg, T.; Andersen, E.; Bastert, O.; Berthold, T.; Bixby, R.E.; Danna, E.; Gamrath, G.; Gleixner, A.M.; Heinz, S.; et al. MIPLIB 2010. *Math. Prog. Comp.* **2011**, *3*, 103–163.
44. Achterberg, T.; Koch, T.; Martin, A. MIPLIB 2003. *Oper. Res. Lett.* **2006**, *34*, 361–372.
45. Mitzenmacher, M.; Upfal, E. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*; Cambridge University Press: New York, 2005.
46. Vastrad, B.; Vastrad, C.; Godavarthi, A.; Chandrashekar, R. Molecular mechanisms underlying gliomas and glioblastoma pathogenesis revealed by by bioinformatics analysis of microarray data. *Mod. Oncol.* **2017**, *34*, 182.
47. Wang, G.; Li, Y.; Zhang, D.; Zhao, S.; Zhang, Q.; Luo, C.; Sun, X.; Zhang, B. CELSR1 acts as and oncogene regulated by miR-199a-5p in glioma. *Cancer Man. Res.* **2020**, *12*, 8857–8865.
48. Baste, J.; Jaffke, L.; Masařík, T.; Philip, G.; Rote, G. FPT algorithms for diverse collections of hitting set. *Algorithms* **2019**, *12*, 254.
49. Baste, J.; Jaffke, L.; Masařík, T.; De Oliveira Oliveira, M.; Philip, G.; Rosamond, F.A. Diversity of solutions: An exploration through the lens of fixed-parameter tractability theory. *Artif. Intell.* **2022**, *30*, 103644.
50. Gao, J.; Goswami, M.; Karthik C. S..; Tsai, M.; Tsai, S.; Yang, H. Obtaining Approximately Optimal and Diverse Solutions via Dispersion. In Proceedings of the LATIN 2022: Theoretical Informatics - 15th Latin American Symposium, Guanajuato, Mexico, November 7-11, 2022, Proceedings; Castañeda, A.; Rodríguez-Henríquez, F., Eds. Springer, 2022, Vol. 13568, *Lecture Notes in Computer Science*, pp. 222–239. https://doi.org/10.1007/978-3-031-20624-5_14.
51. Hanaka, T.; Kiyomi, M.; Kobayashi, Y.; Kobayashi, Y.; Kurita, K.; Otachi, Y. A framework to design approximation algorithms for finding diverse solutions in combinatorial problems. In Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence, 2023, Vol. 37, pp. 3968–3976. https://doi.org/https://doi.org/10.1609/aaai.v37i4.25511.