Self-avoiding multiwalks over tableau-defined neighborhoods in combinatorial optimization: an alternative to ‘nature-inspired’ approaches

Franc Brglez
Computer Science, NC State University
Raleigh, NC 27695, USA
Email: brglez@ncsu.edu

Abstract—Unlike metaheuristic methods that are ‘nature-inspired’, the proposed multiwalk solver adopts two structures from mathematics: (1) a ‘sparse ruler’ with \( m \) marks, and (2) a neighborhood \( \mathcal{N}(m) \) of size \( O(m^2) \) induced by \( m \) mark coordinates. We study neighborhoods \( \mathcal{N}(m) \) in both the binary domain defined by \( 2^L \) binary strings and the permutation domain, defined by \( L! \) comma-separated permutation strings. Each domain induces a vertex-weighted regular sparse graph with either \( 2^L \) or \( L! \) vertices where weights represent values of the respective objective functions \( \Theta(A, \xi) \) or \( \Theta(A, \pi) \). Before making the next step from a vertex \( v_i \), \( O(m) \) bots probe \( L \) ‘neighborhood vertices’ \( v_{i,j} \) in the \( 2^L \)-graph whereas in the \( L! \)-graph, \( O(m) \) bots probe \( L - 1 \) 'neighborhood vertices' \( v_{i,j} \). Under the ‘simple formulation’, solver makes \( O(1 + L) \) or \( O(L) \) evaluations of objective function to probe the pivot and all neighborhood vertices; under the ‘tableau formulation’ introduced in this article, solver makes only \( O(2) \) evaluations of the objective function to probe the pivot and all of the neighborhood vertices. Both the multiwalk and the ‘tableau formulation’ are factors that explain the unmatched scalability for this generation of solvers. The theoretical underpinning for heuristics in the design of these solvers is supported by the model of absorbing Markov chains and its fundamental matrix.

I. INTRODUCTION

Stochastic optimization is the top layer of the ‘Big data analytics’ pyramid [1]. In contrast to the abstract from a 2013 journal article ‘Metaheuristics – the Metaphor Exposed’ [2]:

“... a true tsunami of ‘novel’ metaheuristic methods, most of them based on a metaphor of some natural or man-made process. The behavior of virtually any species of insects, the flow of water, musicians playing together – it seems that no idea is too far-fetched to serve as inspiration to launch yet another metaheuristic. ...”

this article may read as an antidote. We argue for alternative views in the following seven sections and the appendix:

- Section 2: Markov chain formulation generalizes the textbook case of the single ‘drunkard’s walk’ on a plateau to uphill-downhill multiwalks across landscapes defined by sparse vertex-weighted regular graphs, induced by binary or permutation coordinates.
- Sections 3, 4: Matrix permanent and Linear ordering problems induce their landscapes with permutation coordinates of length \( L \) and formulate two tableau schemas to evaluate \( L - 1 \) neighbors of each vertex in time \( O(2) \).
- Section 5: Maximum satisfiability problem induces its landscape with binary coordinates of length \( L \) and formulates a tableau schema to evaluate \( L \) neighbors of each vertex in time \( O(2) \).
- Section 6: Beautiful evidence summarizes results of experiments with three combinatorial solvers that significantly outperform the state-of-the-art ‘nature-inspired’ solvers. The summary relies on asymptotic prediction of a combinatorial solver runtime complexity in the form \( y = a \times b^L \): each data point represents the runtime cost as the mean value of first-passage time with sample size \( \geq 100 \).
- Section 7: Ongoing and future work is an invitation to participate in this research: rapid prototypes in \( \mathbb{R} \) on a number of tableau schemas to evaluate graph neighbors in time \( O(2) \) are ready-to-share, ranging from minimum vertex cover to maximum clique, set packing, and beyond.

Appendix: First passage time statistics analyzes stochastic experiments under two stopping criteria:

1. specify runtime limit in advance and report the best value returned by solver;
2. record runtime when solver reaches the best-known-value (the target value) for the first time – i.e. the first-passage time.

The solvers referenced in Section 6 are not the only one that outperform the state-o-the-art ‘nature-inspired’ solvers. The principal reason why our solvers, including the ones in [6], [7], [8], [9] outperform the state-o-the-art ‘nature-inspired’ solvers even before implementing a multiwalk is this: the first-passage time stopping criterion provides a reliable tool to gauge the true significance of ‘improvements’ that are being measured during the design of any stochastic solver.

In [6] we used the first-passage time stopping criterion by another name to demonstrate that short self-avoiding walks with random restarts outperform a known evolutionary algorithm. In [7], we introduce self-avoiding walks on graphs formed by concatenation of ternary and binary coordinates to outperform a 2005-version of a genetic algorithm, predicting protein tertiary structures in the 2D HP model. In [3], an asymptotic complexity experiment outperforms tabu-search, memetic-search, and a number of alternative solvers on Paley tournament graphs. In [4] ‘memetic algorithm’ applied to the hard problem of low autocorrelation binary sequence performs no better than a random search – the computational part that is effective is the tabu-based search which is outperformed by search based on the self-avoiding walk. In [5], experiments demonstrate 2-orders of magnitude improvement over a 2015-version of a genetic algorithm when searching for an optimum Golomb ruler. In [9], we achieve a significant and consistent reduction in the convergence to the known-best-values for a number of hard test instances – when compared with
convergence rates returned by seven pre-tested configurations of a state-of-the-art Differential Evolution solver.

**Coordinate neighborhoods** \( N(m) \) of size \( O(m^2) \). In [9] we introduce a multiwalk stochastic algorithm searching for global minima in \( \mathbb{R}^p \). This algorithm adopts two structures from mathematics: (1) a ‘sparse ruler’ with \( m \) marks [10], and (2) a neighborhood of \( m \) mark coordinates \( N(m) \). A ruler, where all \( m \) marks are integer coordinates in positions such that the number of unique coordinates in \( N(m) \) reaches \( m(m-1)/2 \), is known as a ‘Golomb ruler’ [11]. When \( m \) coordinates are defined as permutations, binary, ternary, n-ary strings, etc, unit distances between such coordinates induce neighborhoods \( N(m) \) with an upper bound of order \( O(m^2) \).

**Multiwalks under binary/permutation coordinates.** Multiwalks in Figures 2, 3, and 4, serve only to illustrate the viability of the first prototypes under both binary and permutation coordinates. Each stochastic solver uses a multiwalk strategy, evaluating either \( \Theta(A, \xi) \) or \( \Theta(A, \pi) \):

- assign \( O(m) \) initial pivot coordinates to \( O(m) \) bots;
- let each bot probe of the entire pivot neighborhood, concurrently with other bots, and make the best selection for the next step;
- let each bot make the next self-avoiding step using the best selection from the current neighborhood probe;
- terminate the walk for all bots as soon as one of the bots reaches the best-known-value (BKV) of the objective function.

Variations on this strategy are still evolving, including where to initiate and when/where to restart \( O(m) \) bots.

**II. Markov Chain Formulation**

The absorbing Markov chain and its fundamental matrix provide a powerful model to analyze problems in combinatorial optimization. This article relies on notation and concepts introduced in [12] that bypasses the theory of eigenvalues as discussed in the seminal article by Kac [13], the first to formulate and analyze random walks on graphs.

The drunkard’s walk is a textbook example of an absorbing Markov chain, see for example [14]. In our context, we consider an \( L \)-vertex path graph with \( (L-1) \) edges, with each edge implicitly bidirectional – with exception of \( L \)-th vertex having a self-loop edge and only an incoming edge from vertex \( L-1 \). When transformed to a state-transition probability matrix, the state corresponding to this vertex is called an ‘absorbing state’ and we call this vertex an ‘absorbing vertex’.

We generalize the graph in drunkard’s walk to a sparse \( k \)-regular vertex-weighted graphs with \( L \) vertices and \( k \) \& \( L \) edges. We designate \( L \)-th vertex as the absorbing vertex, removing the \( k \) outgoing edges from vertex \( L \) while adding a self-loop edge to it. The \( 3 \)-regular \( 8 \)-vertex weighted graph example in Figure 1 supports a number of definitions and observations:

- **(a)** \( 3 \)-variable, \( 9 \)-clause maxsat instance in conjunctive normal form (cnf) and its objective function \( \Theta(A, \xi) \).
- **(b)** landscape table in a canonical order. The most important columns are coord and value with coord standing in for coordinate and value referring to the value of the objective function \( \Theta(A, \xi) \) being evaluated for the given coordinate \( \xi \). A binary coordinate is a natural choice here. The objective function in our formulation counts the number of clauses that are ‘not satisfied’ for a given coordinate. For example, there are 4 unsatisfied clauses for \( 001 \) while there a 0 unsatisfied clauses for \( 100 \).

The value of index is computed directly from coord, the value of rank represents the number of 1’s in coord, state stands in for a unique name of each coord.

The canonical order implies a partition of the landscape table into non-absorbing or transient and absorbing states: all rows where the column value achieves a minimum (0 in this example) are placed as the last rows in the table, defining the absorbing states. This partition is critical to formulating the absorbing Markov chain and the partitions in its fundamental matrix [14].

- **(c)** value-induced landscape graph associates each pair in the coord/value columns with a weighted vertex placed onto a grid with edges are induced by unit Hamming distances between the coordinates. visually, each vertex suggests Had values associated with vertices \( A, B, C \) been assigned the value of 3, we would re-label the vertex \( C \) as a plateau. As we shall see shortly, such labeling plays a major role in transforming its adjacency graph data structure to a state-transition probability matrix.

- **(d)** coordinate-induced landscape graph is an isomorph of the value-induced graph; the placement of vertices on the \( x \)-axis is strictly in the order of increasing rank of its coordinates; the goal of the placement with respect to the \( y \)-axis is only to achieve an appearance of uniform spacing between vertices at each value of rank. The maximum rank of this weighted \( 2^L \)-vertex very sparse graph is \( L \). The notion of ‘coordinate rank’ is universal in our approach: binary coordinates follow a binomial distribution with respect to their rank where as permutation coordinates, introduced in the next two sections, follow a Mahonian distribution.

The short default name for this graph is landscape graph and it can be considered a generalization of the Hasse Diagram. This vertex placement is also useful to depict multiwalks in graphs. In this example we have four walks with lengths of 1, 2, and 3. These walks are not random: in this example only, each walk follows a greedy descent naturally to reach the absorbing (target) vertex.

- **(e)** The state-transition probability matrix (stpm) of a fair (unbiased) \( 8 \)-sided dice in the canonical form: the state \( D_0 \) represents the designated absorbing state. Without the theory of markov chains, we can deduce that the number of throws to reach \( D_0 \) will have a geometric distribution with a mean value of 8.

- **(f)** coordinate-induced landscape weighted graph adjacency matrix as shown here has a special form: rather than traditional non-weighted 0/1 matrix, all 1’s are replaced with the weight of each vertex and all 0’s are replaced
Fig. 1: (a) 3-variable, 9-clause maxsat instance and objective function $\Theta(A,\xi)$, (b) $\Theta(A,\xi)$ landscape table in a canonical order, (c) value-induced landscape graph, (d) coordinate-induced landscape graph and four walks, (e) state-transition probability matrix (stpm) of an unbiased 8-sided dice, (f) coordinate-induced landscape weighted graph adjacency matrix (adjm), (g) state-transition probability matrix induced by adjm above, (h) walkLength predictions via absorbing markov chain analysis and simulations.

The absorbing Markov chain analysis of the state-transition matrix in canonical form requires a matrix inversion to predict the probability of reaching the absorbing (target state) from any starting state. Intuitively, the expected walkLength from state $A$ is indeed the shortest at 2.3 steps, while 4.5 steps are expected from the state $D$. A sparse, non-uniform distribution of transition probabilities such as shown in (f) induces a wide range of variations in the expected walkLengths. In contrast, the expected walkLength in a dense uniform distribution of transition probabilities such as shown in (e) remains at 8, regardless of the starting state. Moreover, by removing self-loops for all but the absorbing state $D$ reduces the expected walkLength to 7, regardless of the starting state.

A stand-alone article with more details on topics in this section is under construction. The main purpose of Figure 1 is to provide a context and a foundation to support landscape depictions and tableau-based computational schemas for three hard combinatorial problems in the remainder of this article.

### III. MATRIX PERMANENT PROBLEM

In linear algebra, the permanent of a square matrix is a function of the matrix similar to the determinant. Computing a permanent is significantly more difficult than computing a determinant. Moreover, computing the permanent of a $(0,1)$-matrix is #P-complete. It continues to be studied in a number of contexts: from approximations to counting using Markov
The permanent of a matrix $A$ can be defined as:

$$\Theta(A, \pi)_{\text{perm}} = \sum_{\pi_i \in \{1, \ldots, L\}} \prod_{i=1}^{L} a_{\pi_i}(L_{i}(j))$$

where $A$ is a 4x4 matrix instance, and $\pi$ is an edge-weighted bipartite graph. The objective function $\Theta(A, \pi)$ exhibits several properties:

1. **Mismatch numbers**
2. Landscape in a canonical order
3. Coordinate-induced landscape graph and three walks to compute the permanent
4. Pivot coordinate and simple evaluations of $L = 3$ neighbors ($O, G, F$): total cost = $O(L)$
5. Pivot coordinate and tableau evaluations of $L = 3$ neighbors ($O, G, F$): total cost = $O(2)$
6. Four multiwalks in a 120-vertex landscape graph, induced by a 5x5 matrix instance.

While the exhaustive enumeration in (b) that computes the permanent does not scale to large problems, the small example in (a) establishes notation and a routine not only to evaluate pivot vertex at each step of the walk, but also to probe all

**Fig. 2:** (a) definition of matrix permanent, a 4x4 matrix instance, and its equivalent edge-weighted bipartite graph; (b) exhaustive enumeration of primary objective function $\Theta(A, \pi)$: mismatch numbers, (c) objective function $\Theta(A, \pi)$ landscape in a canonical order, (d) coordinate-induced landscape graph and three walks to compute the permanent, (e) pivot coordinate $I$ and **simple** evaluations of $L = 3$ neighbors ($O, G, F$): total cost = $O(L)$, (f) pivot coordinate $I$ and **tableau** evaluations of $L = 3$ neighbors ($O, G, F$): total cost = $O(2)$, (g) four multiwalks in a 120-vertex landscape graph, induced by a 5x5 matrix instance.

The example in Figure 2 has been created to support a number of definitions and observations:

(a) a 4x4 matrix instance, and its equivalent edge-weighted bipartite graph,
(b) exhaustive enumeration of primary objective function $\Theta(A, \pi)$: mismatch numbers,
(c) objective function $\Theta(A, \pi)$ landscape in a canonical order,
(d) coordinate-induced landscape graph and three walks to compute the permanent,
(e) pivot coordinate $I$ and **simple** evaluations of $L = 3$ neighbors ($O, G, F$): total cost = $O(L)$,
(f) pivot coordinate $I$ and **tableau** evaluations of $L = 3$ neighbors ($O, G, F$): total cost = $O(2)$,
(g) four multiwalks in a 120-vertex landscape graph, induced by a 5x5 matrix instance.

While the exhaustive enumeration in (b) that computes the permanent does not scale to large problems, the small example in (a) establishes notation and a routine not only to evaluate pivot vertex at each step of the walk, but also to probe all...
neighbors of the pivot vertex. Since the landscape graph has been induced by permutation coordinates of length \( L = 4 \), each pivot vertex has \( L - 1 = 3 \) neighbors. The landscape table in (c) lists all \( L! = 24 \) coordinates. By designating \( I \) as the pivot vertex with coordinate \( 2,3,1,4 \), we find the three neighbor vertices \( (O, G, F) \) by three pairwise swaps of coordinates with respect to the pivot coordinate. The rank of a permutation coordinate is defined as the minimum number of pairwise swaps to reach the coordinate with the permutation index of 0:1,2,3,4. The rank of the pivot coordinate \( I \) is thus 3 and any of its neighbors will have the rank of \( 3 \pm 1 \).

Before enumerating the matrix for the value of the permanent, we compute the mismatch number \( \mu \) associated with each coordinate. When \( \mu > 0 \) we avoid computing the product matrix elements for the given coordinate. An efficient tableau-based computational schema is illustrated in (f). The row permutation of pivot \( I \) \( 2,3,1,4 \) leads to evaluation of the matrix elements as follows: \( 2,1 \mapsto 0, 3,2 \mapsto 6, 1,3 \mapsto 0, 4,4 \mapsto 0 \). Since only the second element is 6 > 0, incidence vector for this pivot is \((0 1 0 0)\). By computing the weight of the coordinate vector as the sum its values, we get the mismatch number as \( L - \text{weight} \) or \( 4 - 1 = 3 \). To evaluate the mismatch number for the neighbor vertex \( G \), we swap only one pair in the coordinate: from \( 2,3,1,4 \) \( \mapsto \) \( 2,1,3,4 \) and then evaluate the effect of this swap by computing \( 1,2 \mapsto 0 \) and \( 3,3 \mapsto 0 \). Since we saved \((0 1 0 0)\) as the incidence vector of the pivot, we only need to overwrite positions \( 2 \) and \( 3 \), which results in \((0 0 0 0)\) as the incidence vector for \( G \); its mismatch number is then \( 4 - 0 = 4 \). Clearly, under the tableau method in (f), the total cost of computing the mismatch number of pivot and all of its neighbors is \( O(2) \) – a significant reduction when compared to the total cost \( O(L) \) of the simple method in (e).

IV. LINEAR ORDERING PROBLEM

The linear ordering problem arises in a large number of applications and fields such as economy, sociology, graph theory, archaeology, and task scheduling. It considers triangularization of input-output matrices of an economy and the stratification problem in archaeology, where we search for the most probable chronological order of samples from different sites [19], [20], [21].

The example in Figure 2 has been created to support a number of definitions and observations:

(a) a 4x4 matrix instance and objective function formulation \( \Theta(A, \pi) \) of the linear ordering problem,
(b) exhaustive enumeration of the objective function \( \Theta(A, \pi) \): negative total sum above the diagonal,
(c) objective function \( \Theta(A, \pi) \) landscape table,
(d) coordinate-induced landscape graph and two walks that reach the unique minimum of -18,
(e) pivot coordinate \( M \) and **simple** evaluations of \( L = 3 \) neighbors \((E, Q, I)\): total cost = \((O(L^2/2 + (L - 1)L^2/2)\),
(f) pivot coordinate \( M \) and **tableau** evaluations of \( L = 3 \) neighbors \((E, Q, I)\): total cost = \( O(2L^2/2) \).

The small example in (a) establishes notation and a routine not only to evaluate the pivot vertex at each step of the walk, but also to probe all neighbors of the pivot vertex. Under the tableau-based schema in (f) we perform summation of matrix elements above the diagonal for the permutation coordinate \( 1,3,4,2 \) of pivot \( M \) producing the summation vector \((11, 0, 0)\).

\[
\begin{array}{cccc}
M & 3 & 1 & 4 \\
3: & -3 & 1 & 6 & 3.4 & 0 & 3.2 & 5 & = 11 \\
1: & - & 1.4 & 0 & 1.2 & 0 & = 0 \\
4: & - & 4.2 & 0 & = 0 \\
2: & - & & & & &
\end{array}
\]

Vertex \( E \) is a neighbor of pivot vertex \( M \) with permutation coordinate \( 1,3,4,2 \) (we swapped the pair 3,1). Now, we only calculate two summation vector for two rows only:

\[
\begin{array}{cccc}
E & 1 & 3 & 4 & 2 \\
1: & - & 1.3 & 0 & 1.4 & 1 & 1.2 & 0 & = 1 \\
3: & - & 3.4 & 0 & 3.2 & 5 & 5 & = 5 \\
4: & - & & & & & & &
\end{array}
\]

The value of objective function \( \Theta(A, \pi) \) is computed by taking the negative sum all elements in the summation vector. We save the summation vector of the pivot and only update its entries with values returned from each of the neighbors. Total cost of evaluating the pivot and all of its neighbors under the tableau method in (f) is thus \( O(2L^2/2) \) – a significant reduction from \( O(L^2/2 + (L - 1)L^2/2) \) under the simple method in (e).

V. MAXIMUM SATISFIABILITY PROBLEM

The Boolean satisfiability problem (sat) is a fundamental problem in computer science. It arises in a large number of contexts, ranging from artificial intelligence, circuit design, and theorem proving. The maximum satisfiability problem (maxsat) is the problem of determining the maximum number of clauses, of a given Boolean formula in conjunctive normal form, that can be made true by an assignment of truth values to the variables of the formula. It is a generalization of the Boolean satisfiability problem, which asks whether there exists a truth assignment that makes all clauses true. This section provides a new foundation for the design of a stochastic maxsat solver that will not only test the complexity of probabilistic algorithms such as [22] but also compare to computational efficiency of exact sat solvers in the spirit of experiments in [23].

The example in Figure 4 has been created to support a number of definitions and observations:

(a) 5-variable, 6-clause maxsat instance in conjunctive normal form (cnf) and its objective function \( \Theta(A, \xi) \).
(b) objective function \( \Theta(A, \xi) \) landscape table,
(c) coordinate-induced landscape graph and six self-avoiding walks reaching five minima,
(d) pivot coordinate \( E \) and **simple** evaluations of \( L = 5 \) neighbors \((L, \cdot, M, A, G, F)\): total cost = \( O(1 + L) \).
Fig. 3: (a) a 4x4 matrix instance and objective function formulation \( \Theta(A, \pi) \) of the linear ordering problem, (b) exhaustive enumeration of the objective function (OF), (c) landscape table in a canonical order, (d) pivot coordinate \( M \) and **tableau** evaluations of \( L = 3 \) neighbors \((E, Q, I)\): total cost = \( O(2L^2/2 + (L-1)L^2/2) \), (e) pivot coordinate \( M \) and **tableau** evaluations of \( L = 5 \) neighbors \((E, M, A, G, F)\): total cost = \( O(2) \).

The small example in (a) establishes notation and a routine not only to evaluate the pivot vertex at each step of the walk, but also to probe all neighbors of the pivot vertex. Since the landscape graph has been induced by binary coordinates of length \( L = 5 \), each pivot vertex has \( L = 5 \) neighbors. The landscape table in (c) lists all \( 2L = 32 \) coordinates. By designating \( E \) as the pivot vertex with coordinate 00100, we find the five neighbor vertices \((L, M, A, G, F)\) by five flips of coordinate bits with respect to the pivot coordinate. The rank of a binary coordinate is defined as the minimum number of bit flips to reach the coordinate with the binary index of 0: 00000. The rank of the pivot coordinate \( E \) is thus 2 and any of its neighbors will have the rank of \( 1 \pm 1 \).

The objective function \( \Theta(A, \xi) \) counts the number of clauses that are 'not satisfied' for a given coordinate. Our counting uses an efficient tableau-based computational schema in (e).

The title of this section is borrowed from the book [24] that inspired the style of figures we present in this article. Most of the figures and tables attempt to follow the guidelines in this book. In contrast, there are many peer-reviewed publications that report on comparative performance of experiments that are hard to decode and raise questions about their reliability. We briefly discuss reliability issues in the Appendix on First passage time statistics. For additional context, see [25], [26]. The articles from [27] include First Passage Problems in Biology, Exact Record and Order Statistics of Random Walk via First-Passage Ideas, First-Passage Statistics for Random Walks in Bounded Domains, etc.
In theoretical computer science, asymptotic prediction of a combinatorial solver runtime complexity is usually stated in the form:

\[ y = a \times b^L \]

where typically \( a = 1, b < 2 \), and \( L \) is the number of variables in the objective function enumerated by the solver. The notion of runtime is a metaphor that also includes the number of steps taken by walks, the number of function evaluations, etc. For example, a probabilistic algorithm presented in [22] reports asymptotic computational complexity within a polynomial factor of \((4/3)^L\) when the length of all clauses is 3, and the number of variables is \( L \). This is a remarkable achievement.

Now, the question arises: what computational complexity can we observe experimentally for any given solver when applied to a family of 3-sat instances for increasing values of \( L \).

Our experiments with stochastic combinatorial solvers report the asymptotic computational complexity of each solver in similar form

\[ 8.97e^{-4} \times 1.0496^L ; 2.5e^{-5} \times 1.1141^L ; 9.17e^{-6} \times 1.9349^L \]

and are summarized as three case studies in the top part of Figure 5. Notably, all three plots follow the same pattern. The key part of this pattern is an aggregation of at least five data points where each data point represents the first-time passage mean value of runtime cost, based on \( \text{sampleSize} \geq 100 \). Each data point corresponds to a domain-specific problem instance of size \( L \). We select instances from the same 'hardness class' with increasing values of \( L \).

In the bottom part of Figure 5 is a table that lists 6 recent publications, each reporting on results of performance experiments with algorithms. Two items stand out immediately. Five-out-of-six articles show that 0% and 8% of pages are taken to report the results using figures. On the other hand, the number of pages that use tables is as high as 27%; one of the articles spreads a single table sideways across 3 pages! What would
The increasing runtime performance gap between the memetic/tabu search (lssMAts) versus the wandering self-avoiding walk (lssOrel_8), both solving the minimum autocorrelation binary sequence problem on the same platform [4].

The complexity of the meandering self-avoiding walk solving the minimum length Golomb ruler problem. Performance gaps with alternative state-of-the-art solvers are off-the-chart, see [5] for details.

The complexity of the wandering self-avoiding walk solving the optimum linear ordering of Paley tournament graphs. While only a scripted prototype, performance gaps with state-of-the-art alternative solvers are off-the-chart, see [3] for details.

For lessons (not yet) learned, see A Picture is Worth a Thousand Tables [28]

<table>
<thead>
<tr>
<th>citation</th>
<th>pagesTot</th>
<th>tableCnt</th>
<th>tablePages</th>
<th>tablePct</th>
<th>figureCnt</th>
<th>figurePages</th>
<th>figurePct</th>
</tr>
</thead>
<tbody>
<tr>
<td>[29]**</td>
<td>34 6</td>
<td>4.00 19%</td>
<td>0 0 0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[21]</td>
<td>7 4.00</td>
<td>19% 0 0%</td>
<td>1 1 8%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[30]</td>
<td>12 5</td>
<td>1.85 15%</td>
<td>0 0 0%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[31]</td>
<td>13 8</td>
<td>3.20 27%</td>
<td>1 1 8%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[32]</td>
<td>13 8</td>
<td>2.70 9%</td>
<td>12 6.75 24%</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[4]</td>
<td>28 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

** a single table is spread sideways across 3 pages!

Fig. 5: A summary of three asymptotic experiments at-a-glance; all exploring the runtime complexity of the ‘Rosetta stone family’ of hard problems in combinatorial optimization.

be Tufte’s [24] comment on this approach to presentation of experimental results?

VII. ONGOING AND FUTURE WORK

Two articles and supporting prototypes in R are under construction: the first expands on topics of absorbing Markov chain as the model to analyze scaled-down problems in combinatorial optimization. The second expands the repertoire of tableau schemas to efficiently evaluate large graph neighborhoods beyond the ones introduced in this article, ranging from minimum vertex cover to maximum clique, set packing, and beyond. Both articles are advancing current plans to implement uncensored experiments with the C/C++ versions of the three combinatorial optimization solvers introduced in this article. These solvers will extend the range and capabilities of current rapid prototypes in R [33]. Given the expanding repertoire of tableau schemas to evaluate large graph neighborhoods, we invite others to collaborate on these projects.

Acknowledgement. Dr. Rahul Shah and Dr. Tracy Kimbrel suggested to research the problem of matrix permanent.

REFERENCES


\[ \pi = \frac{1}{15} \]

where \( \pi \) is the observed estimate of \( \pi \) under the fixed limit of number of throws, and \( \pi_k \) is the rounded value of \( \pi \) with \( k \) decimal digits.

The notion of first-passage-time is succinctly explained on the back cover of the book *First-Passage Processes* [35]:

In 1777, Buffon formulated and solved the problem of finding the probability that a needle of length \( L \) thrown onto a horizontal plane ruled with parallel straight lines spaced by a distance \( d > L \) will intersect one of these lines. In 1812, Laplace saw this problem in a new light which resulted in a new method of evaluating \( \pi \). He measured the probability of intersection by throwing the needle onto the ruled paper a very large number of times, recording the fraction of throws resulting in an intersection of the needle with a line. In other words, Laplace applied the frequency estimate of probability. Today, the use of a needle and a lined sheet of paper can be considered as the first instance of a stochastic solver that approximates the value of \( \pi \) as reported by Laplace in 1812.

This historical vignette motivated a project to measure, with increasing accuracy, the rate of convergence to 15-decimal digit value of \( \pi_{15} \) by counting the number of throws using four distinct stochastic solvers [34]. The experiments were recorded under two stopping criteria, measuring two types of errors:

**plain:** \[ \epsilon_1 = \hat{\pi} - \pi_{15} \]

**fpt:** \[ \epsilon_2 = \hat{\pi} - \pi_k \]

number of throws is predetermined, report the absolute mean value of the error

where

\[ \hat{\pi} \]

is the observed estimate of \( \pi \) under the fixed limit of number of throws, and

\[ \pi_k \]

is the rounded value of \( \pi \) with \( k \) decimal digits.
Fig. 6: First-passage-time experiments with four stochastic solvers, each approximating the value π to exactly 5 decimal digits. Observe significant variations of the mean values of uncensored throws from solver to solver [34].

... first passage underlies many stochastic processes in which the event, such as a dinner date, a chemical reaction, the firing of a neutron, or the triggering of a stock option relies on a variable reaching a specified value for the first time ...

For a summary of these experiments, see Figure 6. Not unexpected for the sample size of 1100, the target value of π5 = 3.14159, and fpt as the stopping criterion, the solver1 (needles3) stands out as the significantly better solver versus the solver2 (darts): the average of 784.8 throws versus the average of 9445.2 throws.

However, for the same sample size of 1100, the less stringent target value of π3 = 3.142, and plain as the stopping criterion, we cannot tell whether solver1 (needles3) is any better than solver2 (darts). For throws ≥ 10^5, both solvers are returning mean values of ̂π with nearly the same error, regardless of the reference value of πk we use: 3.142, 3.14159, or 3.141592653589793. This experience suggests that we should ask a very different question in order to improve the reliability of comparing the merits of any two stochastic solvers. The question we should ask is this: “what is the mean number of required throws for each solver to reach the same target range?” This very question is being addressed with the first-passage-time stopping criterion fpt.

Typical computational experiments to rank the performance of stochastic optimization solvers are based on a simple approach that is equivalent to stopping criterion plain as defined for four solvers in [34]: take S solvers, P problem instances, N random seeds, run each solver under the stopping criterion of a fixed runtime limit. Then, for each solver, tabulate distances from best-known-values and related statistics. In [36], most insights are revealed in Figure 2 which tallies successes with 18 solvers over all 100 runs for each of 48 objective functions. Here is a verbatim quote: “A ‘success’ was defined as a solution less than 0.005 more than the minimum of the objective function.” In other words, bkv ≤ solution value < bkv + 0.005. Our experiments with this stopping criterion show that solver rankings become increasingly unreliable as the percentage of censored results increases. This observation is also supported with arguments by statisticians [37]. Under criteria defined in Figure 2, the percentage of censored results ranges from (4800 - 3800)/4800 = 21% to (4800 - 1200)/4800 = 75%. If the error tolerance that defines a ‘success’ is reduced from 0.005 to 0.0005, the percentage of censored results in Figure 2 is most likely to increase rapidly towards 100%.

In the context of this appendix, it is appropriate to pay homage to individuals and organizations who in 1950’s provided the underpinning to support the science of reliable experiments: e.g. [38], [39].

Here are four questions asked by Stanley Rasberry[40]:

What one question haunts the best of analytical chemists when their day’s work is done? Four of the main questions that arise regarding any analytical method are:

- Is it sensitive enough for the level of detection required?
- Is it free of interferences for the desired analyte?
- Is it precise, so that the results are reproducible?
- Is it accurate, so that the results approach true values?

These questions can be restated to address reliable performance comparisons of stochastic solvers.