Large W limit of the knapsack problem

Mobolaji Williams

School of Engineering and Applied Sciences,
Harvard University, Cambridge, MA 02138, USA[*]

Jellyfish, Boston, MA 02111, USA[†]

March 31, 2023

Abstract

We formulate the knapsack problem (KP) as a statistical physics system and compute the corresponding partition function as an integral in the complex plane. The introduced formalism allows us to derive three statistical-physics-based algorithms for the KP: one based on the recursive definition of the exact partition function; another based on the large weight limit of that partition function; and a final one based on the zero-temperature limit of the second. Comparing the performances of the algorithms, we find that they do not consistently outperform (in terms of runtime and accuracy) dynamic programming, annealing, or standard greedy algorithms. However, the exact partition function is shown to reproduce the dynamic programming solution to the KP, and the zero-temperature algorithm is shown to produce a greedy solution. Therefore, although dynamic programming and greedy solutions to the KP are conceptually distinct, the statistical physics formalism introduced in this work reveals that the large weight-constraint limit of the former leads to the latter. We conclude by discussing how to extend this formalism in order to obtain more accurate versions of the introduced algorithms and to other similar combinatorial optimization problems.

Keywords: Combinatorial Optimization, Dynamic Programming, Greedy Algorithm, Knapsack Problem, Statistical Physics

[*]williams.mobolaji@gmail.com
[†]mwilliams@jellyfish.co

Contents

1 Introduction

The Knapsack Problem (KP) is a classic problem in combinatorial optimization. In the 0-1 version of the problem [KPP04a], we begin with N objects labeled $i = 1, 2, \ldots, N$ where each object can be included or excluded from a collection. When in the collection, the object i has a value v_i and a weight w_i, and the objective is to find the combination of objects that maximizes the total value while remaining under a given total weight W, called the "weight-limit." A particular collection of objects is defined as $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ with $x_i = 1$ or $x_i = 0$ for object i being included or excluded, respectively, in the collection, and the weight and value vectors are $\mathbf{w} \equiv (w_1, w_2, \ldots, w_N)$ and $\mathbf{v} \equiv (v_1, v_2, \ldots, v_N)$, respectively. Then the objective in solving the KP is to find $\mathbf{x}$ that

$$\text{maximizes } \mathbf{v} \cdot \mathbf{x} \text{ subject to the constraint } \mathbf{w} \cdot \mathbf{x} \leq W\,, \tag{1}$$

where $\mathbf{a} \cdot \mathbf{b} \equiv a_1 b_1 + a_2 b_2 + \cdots + a_N b_N$. For simplicity, we take v_i, w_i, and W to be positive integers.

Among the many standard algorithms for solving the KP [TM90, ros19] there exists a stochastic approach motivated by the physical process of annealing. In the simulated annealing approach to the KP, the negative of the total value of the collection of objects is taken to be the energy of the system, and the system evolves by stochastically sampling the possible objects that can be included in the collection consistent with the weight limit. During this evolution, the temperature parameter is slowly (e.g., logarithmically in time [Ing93]) lowered until the system has settled into its maximum-value collection of objects that is consistent with the weight constraint.

In this work, we analytically model the physical system at the heart of the simulated annealing solution to the KP. Specifically, we compute the thermal partition function for the KP and derive expressions for the average occupancy of each object. By using contour integral identities, we represent an intuitive summation over a discrete state-space as an abstract integration over a continuous contour in the complex plane, and then, by taking the large W limit of the partition function, we transform our discrete-space optimization objective with a constraint into a continuous-space optimization objective with no constraint. The transformation allows us to optimize our problem using analysis thereby simplifying an originally discrete-space analysis. A similarly inspired approach was taken in [And88] by using a mean field approximation and the saddle-point approximation to model the annealing of a spin-system, and here we use similar methods to derive new algorithms for the KP and then use the formalism to understand the relationships between more well-known algorithms.

This work exists at the intersection of combinatorial optimization and statistical physics and reflects the theme of gaining insights into the former by representing them as the latter [MM09, PIM06]. Importantly, this particular intersection makes use of a convenient confluence between statistical physics and combinatorial optimization: While problems in statistical physics tend to get easier as we increase the number of degrees of freedom N (due to the availability of approximation schemes) problems in combinatorial optimization tend to get harder as we increase N (due to increased computation time (Fig. 1)).

The relationship between $N \gg 1$ problems in computer science and $N \gg 1$ limits in statistical physics has been explored in the past [And86, Nis01], but generally the focus of these works for

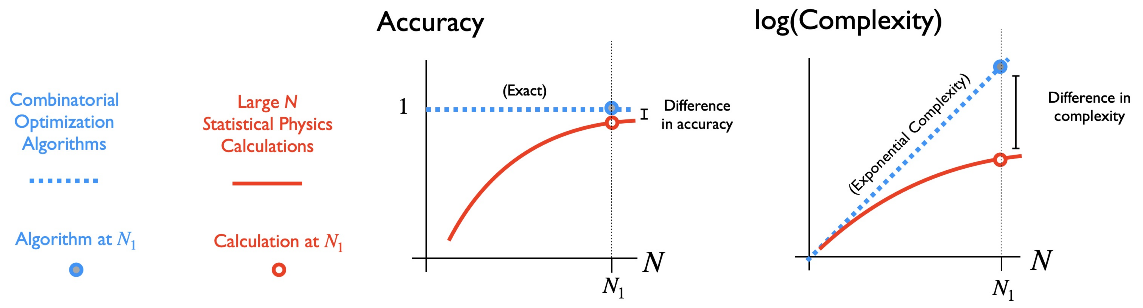


Figure 1: Schematic of accuracy-complexity tradeoff in combinatorial optimization and statistical physics. Combinatorial optimization algorithms are exact, but they often have exponential complexity. $N \gg 1$ statistical physics "algorithms" are approximate but become more accurate as N gets larger, and they have sub-exponential complexity since they are based on evaluating expressions rather than on enumerating an exponential number of states. The relatively quick $N \gg 1$ limit physical analog of a combinatorial optimization problem should be most accurate in the same numerical regime where the exact optimization algorithm takes the longest time to compute.

the KP has been on using annealed approaches or replica methods to study the ground states of physical analogs of combinatorial optimization problems (as in [Fon95, KOL94, ISTO07]) rather than on using statistical physics to find specific solutions to the problems themselves. In the current work, we translate the KP into a physical system with the primary goal of finding a framework for solving the former.

In Section 2, we implement this translation by first deriving the exact partition function for the system. We show that this partition function can be defined through a recurrence relation that reproduces the standard dynamic programming solution to the KP. In Section 3, we approximate the partition function by applying the method of steepest descent in the large W limit. In Section, 4 we derive a solution to the KP from the approximated partition function, and this solution in turn leads to two versions of a statistical-physics based algorithm for the KP. In Section 5, we compare the performances (in terms of runtime and accuracy) of the introduced algorithms to the performances of other standard solutions to the KP for so-called "difficult instances" of the KP [Pis05]. We note that the introduced algorithms do not perform better than standard KP algorithms, but in Section 6 we show that the statistical physics formalism of the new algorithms makes clear relationships amongst the standard ones. In particular, we show that the large W limit of the dynamic programming solution to the KP yields a greedy algorithm in much the same way that the large N limit of the factorial of a number yields Stirling's approximation of said number. We conclude by discussing how higher order corrections could be computed for the introduced algorithms and how the demonstrated connection between dynamic programming and greedy algorithms could be extended to other combinatorial optimization problems.

2 Partition Function for the Knapsack Problem

The partition function is a foundational theoretical construct in statistical physics [McQ73], and if one can calculate it for a system, then all observables for the system can be calculated in turn.

Calculating useful forms of the partition function is difficult for all but the simplest systems [Bax16], but there are often approaches to approximating an answer. For the 0-1 KP considered in the body of the paper, we will derive an expression for the partition function and then use the expression to compute the solution to the KP. Making the calculated results computationally useful will require approximations discussed in the next section, but in this section the results will be exact.

We start by representing the KP as a statistical-physics system at a dimensionless temperature T. This temperature is a non-physical hyperparameter that ultimately be taken to zero to ensure that an optimal solution is found. For the KP, this optimal solution corresponds to the highest-value subset of objects that is consistent with the constraint.

To write the partition function for the KP, we need to place the objective function and the constraint in Eq.(1) in a sum over all possible states $\mathbf{x}$. The possible states consist of all 2^N possible vectors $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ where $x_j \in \{0, 1\}$ and for which only some vectors satisfy the weight constraint. To impose the constraint, we introduce the Heaviside step function $\Theta(j)$ defined as

$$\Theta(j) \equiv \begin{cases} 1 & \text{for } j \geq 0 \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where j is an integer. With Eq.(2) and taking the negative of the total value $\mathbf{v} \cdot \mathbf{x}$ to be the energy of the system, the partition function for the KP is then

$$Z_N(\beta\mathbf{v}, \mathbf{w}, W) = \sum_{\mathbf{x}} \Theta\Big(W - \mathbf{w} \cdot \mathbf{x}\Big) \exp\Big(\beta\mathbf{v} \cdot \mathbf{x}\Big), \tag{3}$$

where $\beta \equiv 1/T$, and the summation over $\mathbf{x}$ is defined as

$$\sum_{\mathbf{x}} \equiv \prod_{j=1}^{N} \sum_{x_j=0}^{1} \qquad \text{[Summation for 0-1 problem]}. \tag{4}$$

On the left hand side of Eq.(3), the subscript N represents the total number of items under consideration for the KP. With the partition function Eq.(3), standard statistical physics formalism tells us that the probability that a particular collection of objects $\bar{\mathbf{x}}$ occurs in the system is

$$P_N(\bar{\mathbf{x}}) = \frac{1}{Z_N} \Theta\Big(W - \mathbf{w} \cdot \bar{\mathbf{x}}\Big) \exp\Big(\beta\mathbf{v} \cdot \bar{\mathbf{x}}\Big). \tag{5}$$

Using this probability to express the average occupancy for object k yields

$$\langle x_k \rangle = \sum_{\mathbf{x}} x_k P_N(\mathbf{x}) = \frac{\partial}{\partial(\beta v_k)} \ln Z_N(\beta\mathbf{v}, \mathbf{w}, W). \tag{6}$$

As an average occupancy for this 0-1 problem, Eq.(6) also represents the probability that object k is included in the collection. For example, if $\langle x_k \rangle > 1/2$, then more than half of the Boltzmann-weighted microstates in the system have object k in the knapsack at a temperature T. Therefore, sampling these microstates would give us a greater than 50% chance of having object k included in the collection.

We can go from an "average occupancy" to an explicit prediction of occupancy by taking averages to the zero temperature limit. For a physical system with a set of microstates $\{\mathcal{S}\}$ and an energy function $-\mathcal{O}(\mathcal{S})$, defined in terms of the positive-definite objective function $\mathcal{O}(\mathcal{S})$, the system partition function (at inverse temperature β) is $Z = \sum_{\{\mathcal{S}\}} \exp\left[\beta\mathcal{O}(\mathcal{S})\right]$, and the average microstate at this temperature is is $\langle\mathcal{S}\rangle \equiv Z^{-1}\sum_{\{\mathcal{S}\}} \mathcal{S}\exp\left[\beta\mathcal{O}(\mathcal{S})\right]$. If there is a single microstate $\mathcal{S}_0$ that maximizes $\mathcal{O}(S)$, then it is easy to show that

$$\lim_{\beta\to\infty}\langle\mathcal{S}\rangle = \mathcal{S}_0. \tag{7}$$

Interpreting Eq.(7) for the KP, the exact solution to the KP is found when Eq.(6) is taken to the $T\to 0$ (or equivalently when $\beta\to\infty$) limit because as temperature goes to zero, the microstate that dominates the summation is that which maximizes $\mathbf{v}\cdot\mathbf{x}$ consistent with the constraint $W \geq \mathbf{w}\cdot\mathbf{x}$. Explicitly, the solution vector $\mathbf{X}$ has components defined as

$$X_k^{\text{soln}} = \lim_{\beta\to\infty}\langle x_k\rangle = \lim_{\beta\to\infty}\frac{\partial}{\partial(\beta v_k)}\ln Z_N(\beta\mathbf{v},\mathbf{w},W). \tag{8}$$

In the quest to establish a statistical physics based KP algorithm, our local goal is to find an expression for the average occupancy Eq.(6) and to use this expression within the zero-temperature solution–or low temperature approximations of the solution–Eq.(8). But to move forward, we first need to express the partition function Eq.(3) in a more mathematically useful form. We do so by moving from a discrete-space summation to a continuous-space integration.

First, we write the Heaviside step function Eq.(2) in terms of a contour integral by using the contour integral expression of the kronecker delta $\delta(j,m)$:

$$\Theta(j) = \sum_{m=0}^{\infty}\delta(j,m) = \frac{1}{2\pi i}\oint_\Gamma \frac{dz}{z^{j+1}}\sum_{m=0}^{\infty} z^m = \frac{1}{2\pi i}\oint_\Gamma \frac{dz}{z^{j+1}}\frac{1}{1-z}, \tag{9}$$

where, in the final equality, we used the geometric series identity $\sum_{n=0}^{\infty} z^n = 1/(1-z)$. The identity is only valid for $|z|<1$, thus applying it constrains the contour Γ to not extend more than 1 unit away from the origin. Now, inserting Eq.(9) into Eq.(3), taking $j \equiv W - \mathbf{w}\cdot\mathbf{x}$, and summing over the states $\mathbf{x}$, we obtain

$$Z_N(\beta\mathbf{v},\mathbf{w},W) = \frac{1}{2\pi i}\oint_\Gamma \frac{dz}{z^{W+1}}\frac{1}{1-z}\prod_{k=1}^{N}\left(1 + z^{w_k}e^{\beta v_k}\right). \tag{10}$$

Eq.(10) is the final form of the KP partition function, and it ultimately allows us to solve the associated problem. To obtain this solution, we make explicit one property that will be useful in subsequent discussions. If we isolate the $k=N$ factor in the integrand of Eq.(10) and expand this factor as two terms, we obtain the identity

$$Z_N(\beta\mathbf{v},\mathbf{w},W) = Z_{N-1}^{(N)}(\beta\mathbf{v},\mathbf{w},W) + e^{\beta v_N} Z_{N-1}^{(N)}(\beta\mathbf{v},\mathbf{w},W-w_N), \tag{11}$$

where $Z_{N-1}^{(k)}(\beta\mathbf{v},\mathbf{w},W)$ is the partition function in which the ℓth component is eliminated from both $\mathbf{v}$ and $\mathbf{w}$, and thus only $N-1$ items are under consideration. Although Eq.(11) was computed

Algorithm 1 Exact Z ($T \neq 0$) algorithm

```
1: Define $\mathbf{w} = (w_1, w_2, \ldots, w_N)$, $\mathbf{v} = (v_1, v_2, \ldots, v_N)$, and $W$. Select a temperature $T$ satisfying $T \ll \min\{v_j\}$.
2: Define an $(N+1) \times (W+1)$ matrix $Z$ with elements set by $Z[i][w] := 1$ for $0 \leq i \leq N$ and $0 \leq w \leq W$.
3: **Computing Partition Function:** Compute $Z[N][W]$ as follows:
4: for $i$ in $[1, N]$ :
5:     for $w$ in $[1, W]$ :
6:         if $w_i > w$:
7:             $Z[i][w] := Z[i-1][w]$
8:         else:
9:             $Z[i][w] := Z[i-1][w] + \exp(v_i/T) Z[i-1, w-w_i]$
10: end
11: **Computuing Solution:** Set $w = W$. Let $\mathbf{X}$ represent the final object occupancy for the KP. Define $\mathbf{X} = (X_1, X_2, \ldots, X_N)$ where $X_j := 0$ initially. The values $X_j$ are then updated as follows:
12: for $j$ in $[N, 1]$ (reverse-order list):
13:     if $1 - Z[j-1][w]/Z[j][w] > 1/2$:
14:         $X_j := 1$
15:         $w := w - w_j$
16: end
```

for $k = N$, the equality applies for any k. In statistical physics, computing the partition function amounts to ”solving” the corresponding system, so Eq.(11) shows that solutions to the KP can be built up recursively in terms of the solutions to instances with smaller weight limits and fewer items. We can write the explicit solution to the KP in terms of this statistical physics representation by applying Eq.(8) to Eq.(10) and using the identity Eq.(11). We then find

$$X_k = 1 - \lim_{\beta \to \infty} \frac{Z_{N-1}^{(k)}(\beta \mathbf{v}, \mathbf{w}, W)}{Z_N(\beta \mathbf{v}, \mathbf{w}, W)}, \tag{12}$$

where $Z_{N-1}^{(k)}(\beta \mathbf{v}, \mathbf{w}, W)$ is the partition function for which the kth component is eliminated from both $\mathbf{v}$ and $\mathbf{w}$. Eq.(12) shows that the explicit solution to the KP can be written in terms of the limit of a ratio of partition functions, and thus being able to compute the partition function Eq.(10) indeed leads to a solution to the KP.

In Algorithm 1, we express the solution Eq.(12) and the partition function recursion Eq.(11) as an algorithm. We call Algorithm 1 an ”Exact Z” algorithm since it is based on computing an exact partition function, but the algorithm itself is an approximate solution to the KP. From the form of Eq.(11) we see that T must be non-zero in order for the partition function to be finite. However, because the exact solution Eq.(12) employs a $T \to 0$ limit, any partition-function-based solution to the KP that uses non-zero T will necessarily be an approximate solution.

Still, we *can* use this formalism to find an exact solution to the KP. We do so by first highlighting a noteworthy aspect of Algorithm 1: The way the partition function matrix $Z[N][W]$ is built up in line 9 is reminscent of the more traditional dynamic programming solution to the KP [KPP04b] in which optimal values for the desired instance are built up from optimal values for subset instances.

To make the connection between the exact partition function Eq.(10) and the standard dynamic programming KP solution explicit, we first compute the average total value of the knapsack, $\langle \mathbf{v} \cdot \mathbf{x} \rangle_{N,W}$, with N items and a weight limit W:

$$\langle \mathbf{v} \cdot \mathbf{x} \rangle_{N,W} = \frac{1}{Z_N(\beta \mathbf{v}, \mathbf{w}, W)} \frac{\partial}{\partial \beta} Z_N(\beta \mathbf{v}, \mathbf{w}, W). \tag{13}$$

Eq.(13) is a temperature-dependent quantity, but we know (from Eq.(7)) that taking the zero-temperature limit of such a quantity yields the optimal value of the quantity across the available states. That is, if $V_N(W)$ is the optimal value for our instance of the KP, then we must have

$$V_N(W) = \lim_{\beta \to \infty} \langle \mathbf{v} \cdot \mathbf{x} \rangle_{N,W}. \tag{14}$$

Now, using Eq.(13) and Eq.(11) in Eq.(14), we find that the recursive relation Eq.(11) leads (see Appendix A) to an analogous recursive relation for $V_N(W)$:

$$V_N(W) = \begin{cases} V_{N-1}(W) & \text{for } W < w_N \\ \max\{V_{N-1}(W),\, v_N + V_{N-1}(W - w_N)\} & \text{for } W \geq w_N, \end{cases} \tag{15}$$

Eq.(15) is the standard dynamic programming solution to the KP [KPP04b], and so we have found that the definition of the KP partition function in Eq.(3) implies the validity of the recursion relation Eq.(11) which itself implies that the optimal value of the knapsack (i.e., the average value at zero temperature) is defined recursively by the standard dynamic programming solution. In essence, the exact KP partition function encodes the dynamic programming solution to the KP.

From this relationship, we see that Eq.(11) and Eq.(15) provide us with two equivalent ways of formulating solutions to the KP. We can either recursively compute the paritition function (and then apply Eq.(12)), or we can recursively compute the optimal value. Both approaches amount to the dynamic programming solution to the KP.

$$\begin{array}{ccc} \text{Standard Dynamic Programming} & & \text{Exact Partition Function} \\ & \cong & \\ \textit{(Implement Eq.(15) recursively to compute } V_N(W)\textit{)} & & \textit{(Implement Eq.(11) recursively to compute } Z_N\textit{)} \end{array} \tag{16}$$

We will revisit this relationship in Sec. 6 when we consider an analogous relationship for the greedy algorithm of the KP.

3 Approximating the Partition Function

We have seen that we can use Eq.(11) to implement a dynamic programming solution to compute the partition function for a given N and W, and then use Eq.(12) to find the occupancy of object j. However, such a computation is typically slower than the standard DP approach due to the computational complexity of the exponential in Eq.(11), especially for large arguments. Another approach could involve computing $Z_N(\beta\mathbf{v},\mathbf{w},W)$ directly from Eq.(3), however, this computation would amount to a brute force solution which requires a summation over all 2^N states of the system. Alternatively, we could try to use Eq.(10) to compute the partition function, but this calculation too would be stymied, this time by the general numerical intractability of the contour integral.

These challenges suggest we explore an alternative form for the KP partition function in order to more fully capture the algorithmic potential of the statistical physics formulation. This alternative form we find will be based on an approximation of the contour integral. Towards finding this approximation, we first write Eq.(10) as

$$Z_N(\beta\mathbf{v},\mathbf{w},W) = \frac{1}{2\pi i}\oint_\Gamma \frac{dz}{z}\,\exp F_N(z;\beta\mathbf{v},\mathbf{w},W) \tag{17}$$

where we defined

$$F_N(z;\beta\mathbf{v},\mathbf{w},W) = -W\ln z - \ln(1-z) + \sum_{k=1}^{N}\ln\left(1+z^{w_k}e^{\beta v_k}\right). \tag{18}$$

It is important to note that Eq.(18) is not defined for all $z\in\mathbb{C}$; given the condition that defines Eq.(9), Eq.(18) is also only valid for $|z|<1$.

To approximate Eq.(17), we use the method of steepest descent [Has13]. The method states that for a given function $f_N:\mathbb{R}\to\mathbb{R}$ that obeys $\lim_{N\to\infty} f_N(x)=\infty$, we have

$$I_N = \int_C dz\,g(z)e^{f_N(z)} = e^{i\theta_1}\sqrt{\frac{2\pi}{|f_N''(z_0)|}}\,g(z_0)\,e^{f_N(z_0)}(1+\epsilon_N), \tag{19}$$

where $g:\mathbb{R}\to\mathbb{R}$; ϵ_N is an error term; C is a contour in the complex plane; z_0 is defined by $f_N'(z_0)=0$; and θ_1 is defined by the constraint $2\theta_1+\theta_2=\pi$ with θ_2 itself defined as the phase of $f_N''(z_0)$ (specifically through $\frac{1}{2}f_N''(z_0)=re^{i\theta_2}$ for $r,\theta_2\in\mathbb{R}$; see [Has13] for the sources of these phases).

In order to apply Eq.(19) to Eq.(17), we need to identify $g(z)$ and $f(z)$. There is some ambiguity in how we make these identifications, but in general for these steepest descent approximations, one subsumes all factors into the exponential argument aside from the $1/z$ factor typical of contour integrals. By this convention, we can identify $1/z$ with $g(z)$ and $F_N(z;\beta\mathbf{v},\mathbf{w},W)$ with $f_N(z)$[1].

Since $|z|<1$, we find that $F_N(z;\beta\mathbf{v},\mathbf{w},W)$ goes to ∞ when $W\to\infty$. Thus we can infer that W is a suitable large-number parameter, and we can apply the steepest descent approximation [Has13] to Eq.(17) for the case of $W\gg w_j$ for all j. Note that taking $W\gg 1$ with no constraint on w_j is not sufficient for ensuring that we are in a regime where the steepest descent approximation applies. An instance satisfying $W\gg 1$ could be trivially created by taking $W\to W'=\lambda W$ and $w_j\to w_j'=\lambda w_j$ for $\lambda\gg 1$, but such an instance would be identical to the original instance defined by W and w_j.

Applying the steepest descent approximation to Eq.(17) for $W\gg w_j$, we obtain

$$Z_N(\beta\mathbf{v},\mathbf{w},W) = \frac{1}{\sqrt{2\pi z_0^2\partial_z^2F_N(z_0)}}\exp F_N(z_0;\beta\mathbf{v},\mathbf{w},W)\left(1+\mathcal{O}(w_j/W)\right), \tag{20}$$

[1]As an alternative, we could include the $1/z$ factor in our definition of $f_N(z)$ (and also take $g(z)=1$) but then the contribution of the $1/z$ factor to determining the critical point z_0 would disappear when we take $W\gg 1$ since $(W+1)\ln z\simeq W\ln z$ in this limit.

where z_0 is the value of z at which $\partial_z F_N(z; \beta\mathbf{v}, \mathbf{w}, W) = 0$. Note that to obtain Eq.(20), we assumed $\partial_z^2 F_N(z_0)$ to be real and positive, thus giving us $\theta_2 = 0$ which in turn implies $\theta_1 = \pi/2$ and $e^{i\theta_1}/i = 1$. Below we verify this assumption. The error term $\mathcal{O}(w_j/W)$ comes from the fact that the error associated with the approximation is of the order of the inverse of the large-number parameter (see equation 11.33 in [Has13] for a full expansion).

In what follows, we will take $F_N(z) \equiv F_N(z; \beta\mathbf{v}, \mathbf{w}, W)$ where notationally convenient. Using Eq.(18), we find that the z derivative of F_N is

$$\partial_z F_N(z; \beta\mathbf{v}, \mathbf{w}, W) = -\frac{W}{z} + \frac{1}{1-z} + \frac{1}{z}\sum_{i=1}^{N} \frac{w_i z^{w_i} e^{\beta v_i}}{1 + z^{w_i} e^{\beta v_i}}. \tag{21}$$

Therefore, the condition that defines z_0 in Eq.(20) is

$$W = \frac{z_0}{1-z_0} + \sum_{i=1}^{N} \frac{w_i z_0^{w_i} e^{\beta v_i}}{1 + z_0^{w_i} e^{\beta v_i}}. \tag{22}$$

We can ensure that such a z_0 always exists as follows. Taking the limits of Eq.(21) at the bounds of the domain $z \in (0, 1)$, we have $\lim_{z\to 0} \partial_z F_N(z) = -\infty$, and $\lim_{z\to 1} \partial_z F_N(z) = +\infty$ which implies that $\partial_z F_N(z)$ crosses the axis at some point between $z = 0$ and $z = 1$. Therefore, by the intermediate value theorem [R+64] there must exist some $z_0 \in (0, 1)$ such that $\partial_z F_N(z_0) = 0$.

Next we check that $\partial_z^2 F_N(z_0)$ is real and positive. Differentiating Eq.(21) and setting $z = z_0$ we find

$$\partial_z^2 F_N(z_0; \beta\mathbf{v}, \mathbf{w}, W) = \frac{1}{z_0(1-z_0)^2} + \frac{1}{z_0^2}\sum_{i=1}^{N} \frac{w_i^2 z_0^{w_i} e^{\beta v_i}}{(1 + z_0^{w_i} e^{\beta v_i})^2} \tag{23}$$

which, for $z_0 \in (0, 1)$, is a positive real number. Thus Eq.(20) is a real quantity, and the z_0 determined from Eq.(22) defines a local minimum for $F_N(z)$.

Finally, we show that z_0 is unique. First assume the contrary, namely that there are two distinct critical points z_A and z_B that satisfy the critical point condition, i.e., $\partial_z F_N(z_A) = \partial_z F_N(z_B) = 0$ and $z_A \neq z_B$. By Eq.(23), we deduce that both of these critical points are local minima. However, if a single-variable function has two local minima at distinct points it must also have a local maximum between those points. Thus, if there are two critical points, there must be a third critical point z_C defining a local maximum. However, we showed in Eq.(23) that if z_C is a critical point of $F_N(z)$, then z_C must be a local minimum. We have found that in order for z_C to exist it must be both a local minimum and a local maximum which is impossible for a single-variable function. Therefore, the initial assumption that there can be two distinct critical points is false, and the critical point z_0 must be unique.

Given that z_0 exists and is unique, we can use Eq.(20) to find a unique approximation to the KP partition function. Such an approximation is useful because it allows us to study the statistical physics of the KP without the summation over the 2^N microstates that defines the initial form of the partition function Eq.(3). In particular, with Eq.(6) and Eq.(8) we know that the solution to the KP can be expressed in terms of derivatives of this partition function, and consequently, an approximate solution to the KP can be expressed in terms of derivatives of the approximate partition function Eq.(20). We pursue this approximate solution in the next section.

4 Solving the Knapsack Problem

Having obtained the unique z_0 determined by the condition Eq.(22), we can use Eq.(6) and Eq.(20) to find an approximate expression for $\langle x_\ell \rangle$ and in turn use this expression to solve the KP. We first consider nonzero-temperature solutions to the KP, and then show how these solutions lead to a zero-temperature approach.

4.1 Nonzero-Temperature

We seek to use the large W approximation results Eq.(20) to explicitly solve the KP. We begin by writing $\langle x_\ell \rangle$ in terms of $F_N(z)$ as

$$\langle x_\ell \rangle = \frac{\partial}{\partial(\beta v_\ell)} \ln Z_N(\beta \mathbf{v}, \mathbf{w}, W) = \frac{\partial}{\partial(\beta v_\ell)} F_N(z; \beta \mathbf{v}, \mathbf{w}, W) + \mathcal{O}(w_j/W), \tag{24}$$

where we ignored derivatives with respect to the prefactors in the approximation because they are sub-leading in our large W limit. Calculating this quantity from Eq.(18), we obtain

$$\langle x_\ell \rangle = \frac{z_0^{w_\ell} e^{\beta v_\ell}}{1 + z_0^{w_\ell} e^{\beta v_\ell}} + \mathcal{O}(w_j/W). \tag{25}$$

Assuming we can find z_0 with Eq.(22), we can use Eq.(25) to determine the average occupancy for the object ℓ. We can then use all of these approximate result as the basis for a large W finite temperature algorithm for the KP. To formulate the algorithm, we assume that we have a black-box solver that can obtain the numerical solution to a nonlinear equation (e.g., [Res91, Com21]). We denote this black-box solver as `NSolve`, and, notationally, we write $x_0 = \texttt{NSolve}(x; F(x))$ when x_0 is the solution the algorithm finds to the equation $F(x) = 0$. As established in the Sec. 3, there is only one solution to Eq.(22), so we do not need to be concerned with multiple roots. The average in Eq.(25) yields a value between 0 and 1, exclusive, and so in order to convert the result into an unambiguous solution we need to introduce a paramater p_{thresh} that defines how large $\langle x_\ell \rangle$ needs to be in order for object ℓ to be included in the collection. In Algorithm 2, we formulate these ideas as a sequence of steps that yields an approximate solution to the KP.

In Fig. 2b, we show the results of applying this algorithm to an example instance. As the temperature T of the system is lowered, the total weight of the included objects predicted from Eq.(29) approaches the limiting weight, and the total value increases to its optimal value. This is as we should expect: In statistical physics, as system temperature T is lowered, the space of microstates where the system spends most of its time gets smaller, until, at zero temperature, the system settles into the single lowest-energy microstate, presuming such a microstate exists.

It is worth comparing Algorithm 2 with simulated annealing, another algorithm extending from statistical physics. In simulated annealing, a typically discrete state space is explored by randomly proposing and then rejecting or accepting state transitions in an energy landscape with gradually

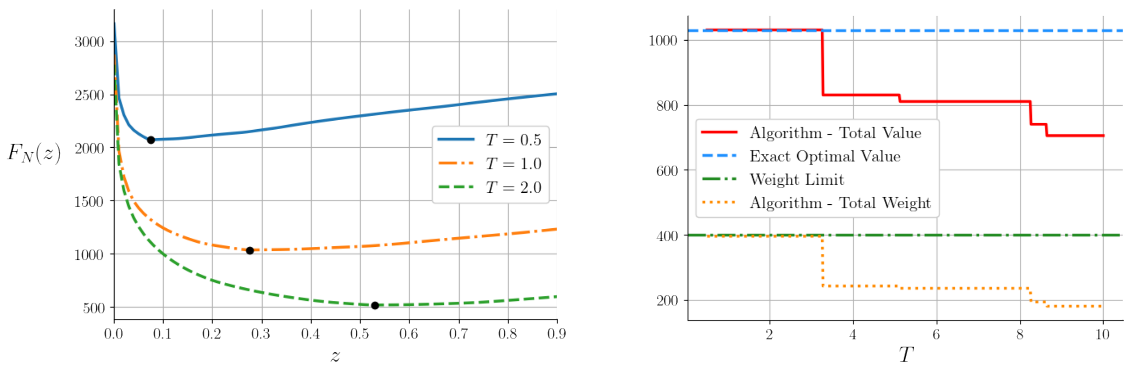
Algorithm 2 Large W nonzero-temperature algorithm

```
1: Define $\mathbf{w} = (w_1, w_2, \ldots, w_N)$, $\mathbf{v} = (v_1, v_2, \ldots, v_N)$, and $W$. Select the system temperature $T$
   satisfying $T \ll \min\{v_\ell\}$, and a probability threshold $p_{\text{thresh}}$.
2: Compute $z_0(T)$ by using a numerical solver NSolve to evaluate
       $z_0 = \texttt{NSolve}(z; G_N(z; \mathbf{v}/T, \mathbf{w}, W)),$   (26)
   where
       $G_N(z; \mathbf{v}/T, \mathbf{w}, W) = -W + \frac{z}{1-z} + \sum_{i=1}^{N} \frac{w_i}{e^{-v_i/T} z^{-w_i} + 1},$   (27)
   at the chosen $T$.
3: Compute $\langle x_\ell \rangle$ from
       $\langle x_\ell \rangle = \frac{1}{e^{-v_\ell/T} z_0^{-w_\ell} + 1}.$   (28)
4: For each $\ell = 1, \ldots, N$ compute $X_\ell$ according to
       $X_\ell = \begin{cases} 1 & \text{if } \langle x_\ell \rangle > p_{\text{thresh}}, \\ 0 & \text{otherwise.} \end{cases}$   (29)
   The vector $\mathbf{X} = (X_1, X_2, \ldots, X_N)$ represents the final object composition.
```

deepening valleys [Ing93]. The system temperature, which is lowered over time, parameterizes the deepening of the valleys, and as these valleys become more pronounced, the system eventually settles into one of its local minima. In this way, simulated annealing, allows us to computationally find local optima in state spaces.

These aspects of simulated annealing have analogs in the large W algorithm. In simulated annealing the objective function is the negative of the total value of the object collection, while in Algorithm 2 the objective function is the more abstract complex potential $F_N(z)$. In simulated annealing, the algorithm takes small random steps in the direction of the the local minimum of the objective function, while in Algorithm 2, solving for z_0 amounts to directly going to the minimum of the corresponding objective function. The two algorithms have consonant interpretations of how the optimization proceeds, even though each is searching different state spaces with different objective functions.

Moreover, the "minimmum-seeking" interpretation of Algorithm 2 suggests a connection to a standard approximation in combinatorial optimization. In finding z_0 (the minimum of $F_N(z)$), we are effectively "rolling down" the potential defined by $F_N(z)$ and settling at its lowest point (See Fig. 2a). In fact, we could rewrite Algorithm 2 as a gradient descent algorithm to highlight this "valley rolling" interpretation. Such an interpretation reveales that the algorithm is essentially a "greedy approach" in z-space for finding the minimum of the potential $F_N(z)$. The KP algorithm itself has a greedy solution, so it is worth asking whether there is any relationship between the well-known greedy heuristic for the KP and the greedy optimization of the complex potential $F_N(z)$ in z-space. Such a relationship would allow us to explicitly connect the exact dynamic programming solution to the KP (shown in Section 2 to be derivable from the partition function) to a greedy solution. We



(a) Plot of Eq.(18) at various temperatures

(b) Calculated value and weight as a function of temperature

Figure 2: Temperature dependence of large W algorithm: Plots correspond to a KP instance with $N = 22$ and a $W = 400$. Exact weights and values are provided in the code referenced in *Data Availability Statement* along with all the functions used to generate the figures. (a) In the $T \neq 0$ algorithm, we are "rolling down" the hill represented by $F_N(z)$ (Eq.(18)) and into the local minimum (marked as black circle). As the temperature of the system is lowered, z_0 decreases. (b) Plot of $\mathbf{v} \cdot \mathbf{X}$ and $\mathbf{w} \cdot \mathbf{X}$ for the X_ℓ computed from Eq.(29) with $p_{\text{thresh}} = 0.95$; As we lower the temperature, the optimized value increases as does the associated weight, until we reach the weight limit.

explore this possibility in Sec. 6.

4.2 Zero-Temperature

In Fig. 2b, we saw that the KP solution given by Eq.(28) became more accurate as system temperature decreased. This behavior suggests a question about the initial solution Eq.(25): Can this solution be taken all the way to $T \to 0$ in order to obtain the best approximation this formalism can give for the KP?

To answer this question, we return to the definition of the KP solution in Eq.(8). This definition shows that we can go from a temperature-dependent average occupancy to the solution of the corresponding optimization problem by taking $\beta \to \infty$. Writing Eq.(25) in terms of the $T \to 0$ limit (for notational convenience), we have

$$X_\ell^{\text{soln}} = \lim_{T \to 0} \frac{1}{e^{-(v_\ell - \gamma(T) w_\ell)/T} + 1} + \mathcal{O}(w_j/W), \tag{30}$$

where we defined

$$\gamma(T) \equiv -T \ln z_0(T), \tag{31}$$

and we wrote $z_0 = z_0(T)$ to make explicit z_0's dependence on temperature T. To use Eq.(30) to obtain a viable solution to the KP, we need to assume that the temperature limit is both non-trivial and well-defined. In other words, we assume that the result is neither zero nor infinite and is instead an explicit function of the arguments v_ℓ, w_ℓ and an implicit function of the parameters that determine $z_0(T)$. The $1/T$ factor in the argument of the exponential of Eq.(30) suggests a final form for the $T \to 0$ limit of Eq.(30). From the definition of the Heaviside step function $H(x)$ with $H(0) = 1/2$, we have

$$H(x) = \lim_{\alpha \to 0} \frac{1}{e^{-x/\alpha} + 1} = \begin{cases} 1 & \text{for } x > 0 \\ 0 & \text{for } x < 0 \end{cases}. \tag{32}$$

Taking the analogous limit in Eq.(30) then gives us

$$X_\ell^{\text{soln}} = H\left(v_\ell/w_\ell - \gamma_0\right) + \mathcal{O}(w_j/W), \tag{33}$$

where we used the identity $H(ax) = H(x)$ for $a > 0$ and defined

$$\gamma_0 \equiv -\lim_{T \to 0} T \ln z_0(T). \tag{34}$$

The process for implementing the solution represented by Eq.(33) is simple, in principle: Given **v**, **w**, and W, we use the constraint Eq.(22) to first determine $z_0(T)$, then determine the function $\gamma(T) = -T \ln z_0(T)$, take this function to the $T \to 0$ limit to obtain γ_0, and finally (by Eq.(33)), include in our KP solution all objects whose v_ℓ/w_ℓ ratio is greater than the value of γ_0.

As a sanity check for this formalism, we will apply it to the trivial "degenerate" KP instance where all N objects have the same weight w_0 and the same value v_0. Such an instance does not have a unique solution, and we expect the derived solution Eq.(33) to reflect this. For the degenerate instance, Eq.(22) simplifies to

$$W = \frac{z_0}{1 - z_0} + \frac{N w_0}{e^{-v_0/T} z_0^{-w_0} + 1}. \tag{35}$$

In order to solve the KP with Eq.(33), we need to determine γ_0. The quantity γ_0 is defined in terms of the zero-temperature limit of $z_0(T)$ and $z_0(T)$ goes to 0 as T goes to zero. Also, the approximation assumes $W \gg w_j \geq 1$. With these two facts, Eq.(35) can be approximated by dropping the first term (which is $O(z_0)$) on the right-hand-side. By dropping this term, Eq.(35) becomes soluble for this degenerate instance, and we find

$$z_0(T) = \frac{W}{N w_0 - W} e^{-v_0/w_0 T} + \mathcal{O}(z_0^2), \tag{36}$$

Computing $\gamma(T)$ from Eq.(31) and Eq.(36) together and then taking the limit of the result as $T \to 0$, we find $\gamma_0 = v_0/w_0$, which by Eq.(33) yields the solution

$$X_i = 1/2, \qquad \text{[Degenerate solution]}. \tag{37}$$

for all i. We expect X_i to be either 1 or 0 to indicate that object i is included or excluded, respectively, from the collection that solves the KP. A value of $1/2$ is therefore ambiguous and such an ambiguity implies that there are multiple viable solutions to the KP for each object: Some solutions where the object is included in the collection and other solutions where the object is excluded from the collection. This structure of solutions is of course true for the degenerate KP since all objects are equivalent and can be switched out of the solution. Therefore the solution Eq.(33) reproduces what

Algorithm 3 Large W zero-temperature algorithm

```
1: Define $\mathbf{w} = (w_1, w_2, \ldots, w_N)$, $\mathbf{v} = (v_1, v_2, \ldots, v_N)$, and $W$.
2: Approximate $\gamma_0$ by using a numerical solver NSolve to evaluate
       $\gamma_0 = \texttt{NSolve}(\gamma; L(\gamma; \mathbf{v}, \mathbf{w}, W)),$   (39)
   where
       $L(\gamma; \mathbf{v}, \mathbf{w}, W) = W - \sum_{j=1}^{N} w_j H(v_j/w_j - \gamma).$   (40)
3: For each $\ell = 1, \ldots, N$ compute $X_\ell$ according to
       $X_\ell = H(v_\ell/w_\ell - \gamma_0),$   (41)
   where $H(x)$ is the heaviside step function with $H(0) = 1/2$. The vector $\mathbf{X} = (X_1, X_2, \ldots, X_N)$
   represents the final object composition. For the case of the ambiguous solution $X_j = 1/2$, we
   take $X_j \to 0$ as an item placement decision.
```

we expect for the degenerate KP[2].

This degenerate instance was special in that we were able to find an analytical expression for $z_0(T)$ for low temperature. However, for non-trivial instances, determining γ_0 from the limit Eq.(34) is difficult due to the need to find low-temperature solutions to Eq.(22), a task which is made difficult due to overflow errors from the exponential function. So instead of determining γ_0 directly from the $T \to 0$ limit of $z_0(T)$, we consider a simpler approach based on the $T \to 0$ limit of the entire expression Eq.(22). Using Eq.(33) and the fact that $\lim_{T\to 0} z_0(T) = 0$, we find that the $T \to 0$ limit of Eq.(22)) is

$$W = \sum_{j=1}^{N} w_j H(v_j/w_j - \gamma_0). \tag{38}$$

Eq.(38) represents a consistency equation for γ_0, and given $\mathbf{v}$, $\mathbf{w}$, and W, we can solve this equation for γ_0. Eq.(38) is not always soluble in this way, and, in such cases, γ_0 would be chosen so as to yield the smallest error in the expression. For example, applying Eq.(38) to the degenerate instance, yields $W/Nw_0 = H(v_0/w_0 - \gamma_0)$ which yields the solution $\gamma_0 = v_0/w_0$ only if $W/Nw_0 = 1/2$. However, regardless of the value of W/Nw_0, the constraint of $0 < W/Nw_0 < 1$ implies that the $\gamma_0 = v_0/w_0$ solution is more consistent with this constraint than other values of γ_0.

In Algorithm 3, we formulate a solution to the KP based on Eq.(38). The algorithm is similar to Algorithm 2 except that rather than seeking the value of z at the potential minimum, we are seeking γ_0 explicitly defined in Eq.(31) and implicitly defined in Eq.(38). From its use in Eq.(38),

[2]The value of $X_\ell = 1/2$ suggests that there is one set of microstate solutions where ℓ is included in the collection and *an equal number* of microstate solutions where ℓ is excluded from the collection. In fact, the exact multiplicity of solutions is a bit different. With $M \equiv \lfloor W/w_0 \rfloor$, we find there are $\binom{N}{M}$ solutions to the degenerate KP of which $\binom{N-1}{M-1}$ include an arbitrary object ℓ and $\binom{N-1}{M}$ do not include that object ℓ. Therefore, the exact statistical value of X_ℓ is actually $\binom{N-1}{M-1}/\binom{N}{M} = M/N$. This value is still a fraction, so it is indeed ambiguous as a representation of whether object ℓ is included in the collection, but it is different from the value of $1/2$ obtained from the large W result Eq.(33). This difference suggests that the value of $X_\ell = 1/2$ only implies "solution ambiguity" and does not give the exact microstate proportionality associated with that ambiguity.

the quantity γ_0 is imbued with a simple conceptual interpretation. With H constrained to be either 1 or 0 (i.e., no ambiguous solutions), it is apparent that Eq.(38) represents a KP solution in which all items are included in the knapsack if their value-to-weight ratios exceed γ_0. Thus, γ_0 is the minimum value-weight ratio that determines knapsack occupancy.

In the next section, we will take all three introduced algorithms and compare them with some standard algorithms for the KP. The goal in this comparison is to evaluate the claim made in Fig. 1 and to see whether a statistical-physics based approach to the KP yields solutions with higher accuracy for larger instances while taking comparatively less time than exact algorithms.

In this work, we have only considered the 0-1 KP, but there are many variations to this classic case. In Appendix B we consider three variations (the bounded, unbounded and continuous KPs) and show how the statistical physics partition function can be computed for each one. For the bounded and unbounded problems, we also show how this partition function can lead to large W algorithms for the KP.

5 Runtime and Accuracy Comparisons

In the introduction, we argued that the large-number limit of the KP should (as the large-number parameter increased) yield solutions that are progressively more accurate while taking relatively less time than an exact algorithm based on brute-search or dynamic programming. In this section, we explore whether this is the case by comparing the introduced algorithms to standard KP algorithms.

Three new algorithms have been introduced in this work: The "exact Z algorithm" (Algorithm 1) based on a calculation of the KP partition function through the recursive definition Eq.(11); the "large W nonzero-temperature algorithm" (Algorithm 2) based on the sigmoidal solution for $\langle x_\ell \rangle$ in Eq.(25); the "large W zero-temperature algorithm" (Algorithm 3) based on the Heaviside step function solution Eq.(33). Comparing the runtime and accuracy performances of these algorithms would make evident the relative benefits of the higher accuracy of the exact algorithm versus the faster runtimes of the approximate algorithms in addition to the effect of the $T \to 0$ limit on the accuracy of the approximate algorithm.

Three other standard KP algorithms were used as baseline comparisons. The dynamic programming (DP) algorithm for the KP that has time complexity $O(NW)$; the greedy algorithm which arranges objects as a decreasing sequence in v_i/w_i and then includes objects in the collection according to this sequence until the weight limit is reached [Dan57]; the simulated annealing algorithm in which the KP is represented as a thermal system whose temperature is gradually lowered until the system settles into the microstate of highest total value consistent with the constraint [Čer85].

All six algorithms were applied to four different types of "difficult" KP instances taken from [Pis05]. In [Pis05] it was noted that the easiest instances of the KP consist of those for which item values are uncorrelated with item weights since such instances very likely contain "obviously-included" items with large values and small weights. A follow-up study [SKL+13] further clarified that difficult KP instances occur when there is a strong correlation between the values and weights or when there are degeneracies in the values and weights of the list of items. In this section we show the results of applying the six algorithms to the ”circle” and ”spanner” instances, instances which variously exhibix strong correlations or degeneracies. The results for two other difficult instances from [Pis05]–the ”profit-ceiling instance” and the ”multiple-strongly correlated items instance”–are discussed in Appendix C.

The circle and spanner instances are constructed as follows.

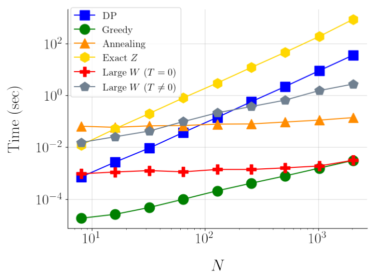
- **Circle Instances:** Values as a function of the weights form the arc of an ellipse. Taking the weights w to be uniformly distributed in the range $[1, R]$, for a free integer paramater R, the values satisfy $v = d\sqrt{4R^2 - (w - 2R)^2}$ where d is also a free parameter. In [Pis05], it is noted that particularly difficult instances come from choosing $d = 2/3$. The value of R was set to 100.

- **Spanner Instances:** All items are multiples of a small set of items termed the ”spanner set.” To create the instance, we first select ν weights w_j in the interval $[1, R]$, for a free integer parameter R, and then define the corresponding values as $v_j = w_j + R/10$. The N items of the instance are found by randomly selecting an item from the ν items in the spanner set, and randomly selecting an integer a from the interval $[1, m]$, for a free integer paramater m. The new item then has the value and weight (av_j, aw_j), and this is repeated until N items are selected. In what follows, we set $\nu = 2$ and $m = 10$, in accordance with the construction in [Pis05]. The value of R was set to 10.

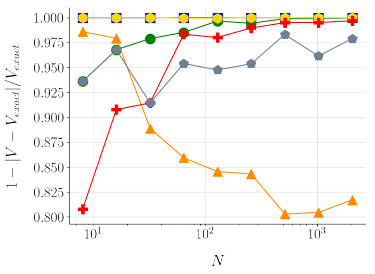
To compare the set of six algorithms, we considered instances with $N = 2^k$ objects for $k = 3, 4, \ldots, 12$. For each N, integer weights were randomly selected and values were calculated according to the instance definitions given above. The weight limit was set to $\sum_{j=1}^{N} w_j/2$ which meant that an increase in N led to an increase in W, with a mostly linear scaling. Having both W and N increase together was important for maintaining the existence of non-trivial solutions: In order to see the effects of large W on the accuracies of solutions, we were primarily interested in considering instances with increasing W, however if W increased without an increase in N, then, given a fixed range of variation for the weights w_j, the instance would eventually admit the trivial solution in which all items are included.

The runtime and accuracy results of applying the algorithms to the two difficult instances are presented in Fig. 3. Fig. 3a, 3b, and 3c displays the results for the circle instance, and in Fig. 3d, 3e, and 3f displays the results for the spanner instance. Fig. 3a and 3d shows the runtime of each algorithm as function of the total number of items. Fig. 3b and 3e shows the accuracy of each algorithm as a function of the total number of items. Fig. 3c and 3f shows the runtime and accuracy for each algorithm for the case of $N = 2048$ items. Since the dynamic programming solution is exact, we defined the accuracy of an algorithm as $1 - |V - V_{\text{exact}}|/V_{\text{exact}}$ where V_{exact} was the optimal value given by dynamic programming, and V was the optimal value given by the particular algorithm.

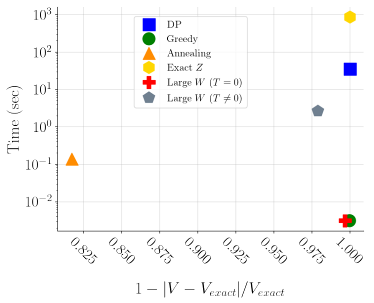
The first noteworthy result is the behavior of the exact Z algorithm. Fig. 3c and Fig. 3f show that the exact Z algorithm (yellow hexagon) is generally a fixed factor slower than the standard dynamic programming algorithm (blue square) for both types of instances, but has the same time scaling with N and achieves similar accuracy (Fig. 3b and Fig. 3e). This reflects the fact that calculating the exact partition function for the KP is tantamount to implementing the standard dynamic



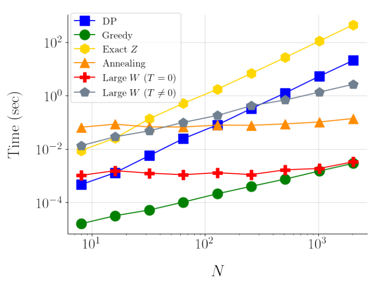
(a) Circle: Time vs. N



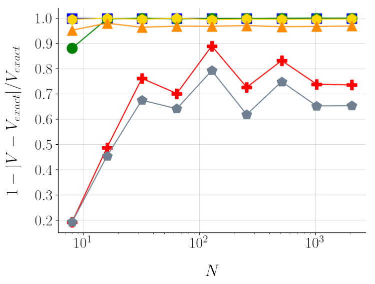
(b) Circle: Acc. vs. N



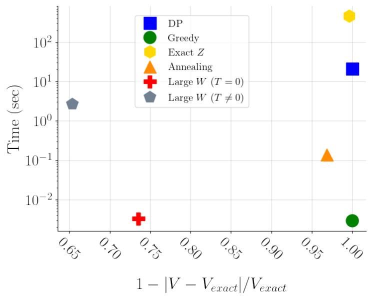
(c) Circle: Time vs. Acc.



(d) Spanner: Time vs. N



(e) Spanner: Acc. vs N



(f) Spanner: Time vs. Acc.

Figure 3: Runtime and Accuracy Comparisons: In (a),(b), and (c), we show the results for the ”circle instances” and in (d), (e), and (f), we show the results for the ”spanner instances.” (a) and (d) depict the runtimes as a function of N. (b) and (e) depict the accuracy of the algorithms as a function of N. (c) and (f) depict the runtime and accuracy for the instance with $N = 2048$ items. For the temperature dependent algorithms we set $T = 1.0$. The large W algorithms can achieve high accuracies for the strongly correlated circle instance but fail for the highly degenerate spanner instance. Code used to generate these figures is linked to in Sec. 12, *Data Availability Statement*.

programming solution to the KP, and thus both algorithms should proceed in $O(NW)$ time with more time required for the exact Z algorithm in order to precisely compute its exponential factor.

The annealing algorithm (orange triangle) performs faster than the dynamic programming and exact Z algorithms but more slowly than both large W algorithms as N increases. This time scaling reflects the fact that the state space annealing must explore grows as 2^N for increasing N and thus takes longer to search for an optimal solution. The annealing algorithm is also less accurate than the algorithms for the strongly correlated (but not degenerate) circle instance, but achieves much better accuracy for the ratio-degenerate spanner instance. This likely reflects the fact that degenerate instances often allow for multiple solutions that an annealing algorithm can converge to, but less degenerate instances require a more extensive (and thus more error prone) search of the state space.

For the large W algorithms, we see that the nonzero-temperature algorithm (grey pentagon) is faster than the dynamic programming solution for values of $N \geq 10^2$. True to the logic of Fig. 1, the faster speed of the nonzero-temperature algorithm stems from the its basis in solving an equation, a process which generally has a lower order scaling with N than exact algorithmic solutions for the KP. The zero-temperature algorithm (red-plus) is even faster than the nonzero-temperature version because while the latter requires time consuming high precision solvers to compute exponential terms at low temperture, the former's constraint equation (Eq.(40)) has simple algebraic factors.

Considering the two large W algorithm's relative accuracies, it is apparent that the nonzero-temperature algorithm is less accurate than the zero-temperature algorithm. We noted in Fig. 2 that the nonzero-temperature algorithm (Algorithm 2) becomes more accurate as T is lowered, and thus nonzero values of T will generally yield solutions that are not yet at their optimal possible values. Thus as the T defining the nonzero-temperature algorithm is lowered, we expect the algorithm's predicted optimal value to converge to that of the zero-temperature algorithm (Algorithm 3), provided there is a means to solve Eq.(27) for progressively lower T.

Although both large W algorithms achieve high accuracies for the circle instance (which has strong correlations between values and weights) neither algorithm achieves high accuracies for the spanner instance (in which the value-to-weight ratio is highly degenerate across items). The reason for the poor performance stems from the properties of the spanner instance. For the chosen construction (with $\nu = 2$ and $m = 10$), there are only two possible values of the ratio v_j/w_j. The zero-temperature algorithm (Algorithm 3) is blind to differences in values and weights where the ratio is the same, and accepts (or rejects) all items that fall above (or below) a given ratio. When there are two distinct ratios, the algorithm only has three choices of how to build up a solution: Accept all items with both ratios, accept all items of only one ratio and reject all items of the other, or reject all items with both ratios. Therefore, the algorithm does not exhibit the selectivity required to build up an optimal solution. This constrained set of choices leads to a poor accuracy for the zero-temperature algorithm. Moreover, for the given spanner instance, there are only 20 possible unique choices for (v_j, w_j) meaning that a system with $N \sim 1024$ items is highly-degenerate with on average 50 copies of each type of item. We recall from Eq.(37), that the zero-temperature algorithm yields ambiguous predictions for highly-degenerate instances, and so we would not expect it to perform well for the spanner instance. The nonzero-temperature version of the algorithm represents a softer (i.e., continuous) version of the acceptance criteria that governs the zero-temperature algorithm and thus shares zero-temperature algorithm's limitations.

The greedy algorithm is also dependent on the ratio v_j/w_j, but it is able to avoid the degeneracy problems of the spanner instance and achieve consistently accurate results because the algorithm adds items more selectively. In particular, the algorithm can continue to add items to the solution as long as doing so does not violate the weight constraint. More generally, for both of the instance types, the greedy algorithm performs well for large values of N, eventually (for the largest N) predicting values close to the optimal total value in much less runtime. This might appear strange to those familiar with how greedy algorithms are discussed in KPs. The greedy algorithm is typically touted as a fast but inaccurate way to find an optimal collection of objects because it myopically looks at the next best choice rather than considering more holistic object combinations. However, given the large W limit, the KP instances satisfy $W \gg w_j$ for all j, and thus including sub-optimal items generally does not preclude the inclusion of other items as needed. In other words, in the limit of $W \gg w_j$, the instances act as ”semi-continuous KP” for which we expect the greedy algorithm to perfom well [KPP04c].

Comparing the performances of the three introduced algorithms to those of the standard algorithms for the KP, we see that the large W algorithms generally do not perform better along the given metrics than existing algorithms. Such a result might strike one as indication of wasted effort. The statistical physics formalism has resulted in algorithms that are more complicated but yield no better performance than pedestrian KP algorithms. However, in Section 2, we noted that the exact Z algorithm was related to the dynamic programming solution to the KP. In the current section, given the definition of the standard KP greedy algorithm, we noted that it made use of value-to-weight ratios in a way similar to these ratios use in the zero-temperature algorithm. This relation in turn suggests that the zero-temperature algorithm can be related to a greedy algorithm. The fact that both the exact Z and zero-temperature algorithms stem from a common statistical-physics starting point suggests that the established and conjectured counterparts of these algorithms (i.e., dynamic programming and the greedy algorithm, respectively) could be similarly related. Making this relationship concrete could in turn allow us to use statistical physics to connect distinct algorithmic spaces of combinatorial optimization problems. It is this connection, rather than the performances of the introduced algorithms, that gives the statistical physics perspective its value. We outline this connection more explicitly in the next section.

6 Greedy Algorithm as Large W Limit of Dynamic Programming

In this section, we show how the statistical physics formalism allows us to relate two approaches to solving the KP, ultimately revealing that the recursive DP solution leads to a myopic greedy solution when the problem is taken to the $W \gg w_j$ limit.

First, we review the fundamental question asked by the zero-temperature and standard greedy algorithms. The large W zero-temperature algorithm (Algorithm 3) asks "What is the minimum value-to-weight ratio (i.e., γ_0 defined in Eq.(34)) above which one can accept all the satisfying items and obtain a total weight that satisfies the weight limit?" The standard KP greedy algorithm asks "Which items are included in the knapsack if one arranges all items in decreasing order of their value-to-weight ratios and accepts items in sequence until the knapsack has a total weight that satisfies the weight limit?"

In the greedy algorithm, the ordering and selective acceptance of items yields a minimum ratio for v_ℓ/w_ℓ above which all objects are included knapsack. Conversely, the zero-temperature algorithm seeks to compute a minimum acceptance-ratio as a first step and then to use this ratio as a rule for including items. In essence, the greedy algorithm follows an "ordering and fill-up procedure" while the the zero-temperature algorithm follows a "criteria procedure" for deciding which items are included in the knapsack, but both algorithms make use of a minimum value-to-weight ratio.

From here, we can ask whether the minimum-ratio calculated through the greedy algorithm is the same as the minimum ratio of the zero-temperature algorithm. To answer this question, we first define the minimum-ratio for the greedy algorithm. Following the standard greedy algorithm for the KP [KPP04c], say that the items in the knapsack are sorted so that their value-to-weight ratios are in non-increasing order:

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \geq \cdots \frac{v_N}{w_N}. \tag{42}$$

Items are then added to the knapsack until the weight-limit is violated. We define α as the index of the item where this violation first occurs:

$$\alpha = \min\Big\{k : \sum_{j=1}^{k} w_j > W\Big\}. \tag{43}$$

By Eq.(42), all items j satisfying $v_j/w_j \geq v_\alpha/w_\alpha$ are inclueded in the greedy KP solution[3], and thus v_α/w_α is the desired minimum ratio. For the zero-temperature algorithm, the solution Eq.(41) indicates that γ_0 is the minimum value-to-weight ratio that determines item inclusion for the algorithm. This γ_0 value is found by selecting the γ at which the equation

$$L(\gamma; \mathbf{v}, \mathbf{w}, W) = W - \sum_{j=1}^{N} w_j H(v_j/w_j - \gamma_0) \tag{44}$$

is zero or minimized for $L \geq 0$. From the minimum ratios of the greedy algorithm and the zero-temperature algorithm, we can define the "normalized ratio difference" as

$$\text{norm. ratio diff.} \equiv \frac{|\gamma_0 - v_\alpha/w_\alpha|}{\gamma_0}, \tag{45}$$

which defines the percent difference by which the greedy minimum-ratio differs from the large W zero-temperature minimum-ratio. Computing the values of this difference for the circle and spanner instance, we find (Code for calculation is linked in Section 12, *Supplementary Code*)

$$\text{norm. ratio diff.} < 10^{-15}, \tag{46}$$

suggesting that the two minimum-ratio values are numerically identical.

Thus, it appears that the standard greedy algorithm and the zero-temperature algorithm are implementing the same basic procedures under different framings. Still, whether that framing concerns ordering items and filling up the knapsack or is one based on a criteria, both the standard greedy algorithm and the zero-temperature algorithms reflect the type of local decision making that typifies a greedy algorithm. Both algorithms are seeking to optimize a variable (i.e., total value) by making choices that do not involve a full exploration of the state-space. Thus we can interpret the zero-temperature algorithm as a greedy algorithm.

$$\begin{array}{ccc} \text{Greedy Algorithm} & & \text{Large } W \text{ Zero-Temperature Algorithm} \\ & \cong & \\ (\textit{Fill knapsack up to min } v_\ell/w_\ell \textit{ limit}) & & (\textit{Compute min } v_\ell/w_\ell \textit{ limit, then fill knapsack}) \end{array} \tag{47}$$

In so far as greedy algorithms go, the zero-temperature algorithm is a fairly unsophisticated one, sharing the basic characteristics of the standard greedy solution (namely a value-to-weight ratio and

[3]Note that the converse is not true, namely not all items in the greedy solution must have value-to-weight ratios greater than v_α/w_α. Specifically, just because adding the αth item violates the weight constraint does not mean adding any subsequent item will also violate the constraint.

the fill to the weight limit procedure) without any of that algorithm's discernment. This difference is reflected in the two algorithm's relative performances in Fig. 3 and is explained by the fact that while the zero-temperature solution only includes items that are above a certain value-to-weight ratio, the standard greedy solution can accept items with ratios below this special ratio if doing so does not violate the weight limit.

Having concluded that the zero-temperature algorithm (Algorithm 3) is a greedy algorithm, it appears that we have done a lot of work just to end up in essentially the same (and arguably a worse) place. If the large W limit of the partition function for the KP yields a greedy algorithm, and this algorithm is less consistently accurate than the standard greedy algorithm, then what value does the new algorithm provide to computer scientists who are interested in finding better ways to solve real problems? Moreover, what value does this formalism provide to physicists who have already studied the statistical mechanics of this system as a representation of a disordered system [KOL94, Ino97]? Here we argue that the value in both directions exists in the path we have taken to seemingly end up in the same place we began[Eli43].

Given the partition function identity in Eq.(11) and the prior discussion on how the zero temperature algorithm is a greedy algorithm, the statistical physics formalism suggests that there is a relationship between dynamic programming and greedy algorithms, at least for the KP. In combinatorial optimization, dynamic programming and greedy approaches to solving problems are by definition very different. In dynamic programming, a problem is solved exactly by recursively referring to stored solutions of sub-problems. In greedy algorithms, a (typically approximate) solution is built by selecting whatever choice is best given an initial state. After deriving the identity Eq.(11), we showed that the exact partition function Eq.(10) yielded the dynamic programming solution to the KP, and above, we explained how the zero-temperature large W algorithm was a greedy algorithm. From the fact that we can move from the exact partition function to the approximate partition function by taking the large W limit, in a sense, applying a large W approximation to the dynamic programming approach to the KP yields a greedy algorithm. We represent the relationships between these algorithms and limits in Fig. 4.

The merit in establishing this relationship can be understood through an analogy. One could imagine knowing that the quantity $I_N = \ln(N!)$ can be calculated recursively as $I_N = \ln N + I_{N-1}$ and also knowing that I_N can be approximated for large N as $I_N \simeq N \ln N - N$, but *not* knowing how the approximation relates to the exact result or why the approximation is a good one. To make clear the relationship between the exact and approximate forms of I_N, we could use the formalism of the Gamma function plus Laplace's method (or alternatively just the Euler-Maclaurin formula) to derive the latter from the former and in turn obtain a systematic way to obtain higher-order corrections to the approximation. As an added benefit, the derivation would serve as an example for computations of similar combinatorial factors, thus providing many other connections between exact and approximate combinatorial expressions.

The statistical physics model for the KP achieves something similar for the dynamic programming and greedy solutions to the KP. It shows how the two algorithms are related and serves as an example for how similar ideas could be applied to find relationships between other algorithms. This mapping only arises from the statistical physics framing of the system, and the representation

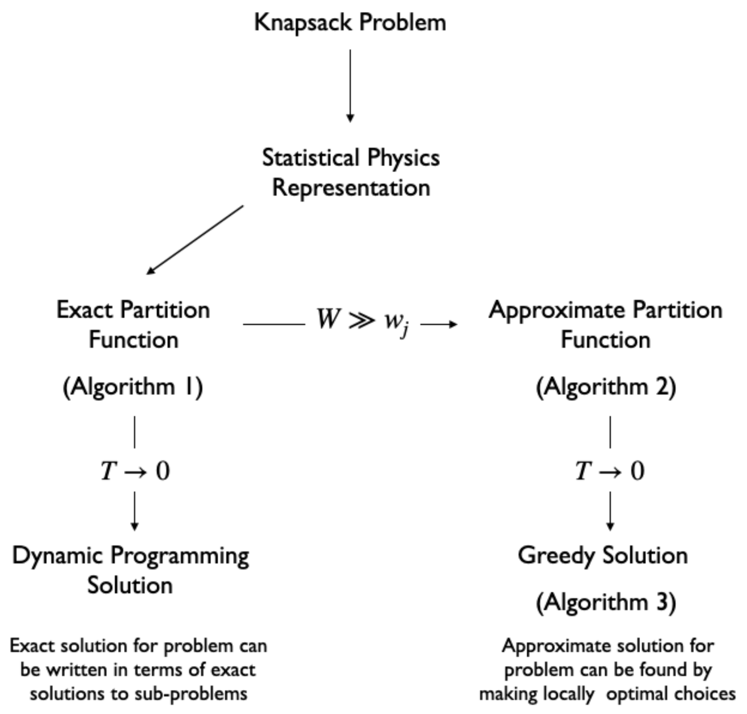

Knapsack Problem
Statistical Physics Representation
Exact Partition Function
(Algorithm 1)
$W \gg w_j$
Approximate Partition Function
(Algorithm 2)
$T \to 0$
$T \to 0$
Dynamic Programming Solution
Greedy Solution
(Algorithm 3)
Exact solution for problem can be written in terms of exact solutions to sub-problems
Approximate solution for problem can be found by making locally optimal choices

Figure 4: Relationship between Dynamic Programming and Greedy Solutions to the KP: For the statistical physics representation of the knapsack problem, taking the $T \to 0$ limit of the exact partition function yields the standard dynamic programming solution, and taking the $T \to 0$ limit of the large-W limit of the partition function yields a greedy solution (Algorithm 3). The parameter T modulates how far away we are from the optimal solution (i.e., lower T corresponds to more optimal solutions). Thus both Algorithm 1 and Algorithm 2 represent, respectively, dynamic and greedy solutions to the KP. In this way, the large W limit of the dynamic programming solution is a greedy solution.

of the algorithms themselves is specific to the KP, but its existence gives us a more comprehensive understanding of how algorithms for the KP relate to one another and motivates further exploration into whether this relationship generalizes for other similarly formulated combinatorial optimization problems.

7 Discussion

Starting from Fig. 1, we began this work by suggesting that the properties of analytical statistical physics made the formalism amenable to solving large N combinatorial optimization problems with high accuracy and in less time than exact solutions. From Fig. 3, this suggestion appears to have supporting evidence in some cases, but it is also clear that already well-known algorithms outperform the ones introduced in this work.

We explored three related algorithms originating from the statistical physics formulation of the problem. The first algorithm (Algorithm 1) followed from a recursive identity for the exact partition function of the system. This algorithm was found to reproduce the standard dynamic programming solution for the KP. Next, by approximating the exact partition function in the large W limit, we obtained an approximate solution to the KP which resulted in another algorithm (Algorithm 2) that was termed the ”nonzero-temperature algorithm.” Finally, by taking this latter solution to the $T \to 0$ limit, we obtained the ”zero-temperature algorithm” (Algorithm 3).

Comparing the accuracies and runtimes of these algorithms to the standard dynamic programming algorithm, the standard greedy algorithm, and simulated annealing, we found that the introduced algorithms did not consistently outperform existing ones for ”difficult instances” (Fig. 3). Algorithm 1 had accuracy results similar to those of the standard DP algorithm, but was a fixed factor slower. Algorithm 2 generally performed worse than Algorithm 3, and while Algorithm 3 did perform better as N increased (with not as much of a runtime increase as that for the exact algorithm), it did not perform well for degenerate KP instances, and the standard greedy algorithm always had better results.

Since well-known algorithms do better in runtime and accuracy than the introduced algorithms, it may seem that the new algorithms have little value. However, the path leading to the approximate algorithms suggests that the value of the statistical physics approach exists outside of what initially motivated it. In particular, the correspondence between the statistical physics algorithms and the more well-known algorithms tells us something about the relationships between the algorithms themselves.

Dynamic programming and greedy solutions to a combinatorial optimization problem are typically seen as distinct and separate ways to solve a common problem. Upon inspecting the two algorithms, it does not appear that they are at all related other than the fact that they use the same parameters. However the organization of Fig. 4 reveals that when the KP is formulated as a statistical physics system, the exact partition function yields the standard dynamic programming solution, and the large W limit of that partition function yields a greedy solution to the KP. Thus, through the statistical physics representation of the KP, we see that taking the large W limit of a dynamic programming solution to the KP yields a greedy solution (Algorithm 3).

In a way, it makes sense that the large W approximation to the KP partition function would yield a greedy algorithm. The large W approximation is based on the method of steepest descent which approximates the partition function by the local minimum of a potential function. The search for this local minimum can be represented as a directed step-by-step movement towards the minimum of the potential, essentially as a greedy search. So the large W approximation already contains a greedy solution within its structure. What is interesting here is that the greedy approach to maximizing the continuous potential is paralleled by a greedy approach for the final discrete KP solution.

There are a few investigations that could naturally follow this work: Understanding error estimates of the solution; finding higher-order corrections to the approximate solutions; applying the formalism to other optimization problems;

The algorithms give us solutions for the KP in the form of the vector $\mathbf{X}$, but they do not give us a sense for how the approximated maximum $\mathbf{v} \cdot \mathbf{X}$ differs from the true maximum. For the exact Z algorithm, such an error estimate could follow from a $T \ll 1$ expansion of the partition function and the argument of the limit in Eq.(12). For the large W algorithms, this error estimate could likely be found by a more careful accounting of the higher order terms in the steepest descent approximation.

These higher-order terms could also serve as the foundation for improved versions of Algorithm 2 and 3. In the standard steepest descent approximation, the potential of the exponential integrand is approximated by a quadratic function. It is this function which yielded the approximate solution Eq.(25) from which the final solutions Eq.(28) and Eq.(41) were derived. By retaining higher-order terms in the integrand expansion, we could obtain more accurate partition functions and in turn more accurate approximate solutions, possibly ones that do not have the all-or-nothing deficiencies of the existing large W algorithms.

Following the formalism that led to Algorithms 1, 2, and 3, it is clear that the ability to relate various limiting forms of the algorithms to each other follows from the representation of the KP partition function as the integral Eq.(10). The integral representation of the KP allowed us to apply the steepest descent approximation in the large W limit, which ultimately connected the dynamic programing representation and the greedy representation of the problem. This integral representation in turn followed from representing the KP weight-constraint as a contour integral. Thus, it would be straightforward to extend this approach to other combinatorial optimization problems with similar types of constraints. In particular, if a problem is restricted by an inequality constraint (leading to a Heaviside function) or an equality constraint (leading to a Kronecker delta function), then much of the prior formalism could be transferred wholesale, and thus repeated application of the formalism to different optimization problems could lead to many correspondence pairs between various dynamic programming and greedy solutions to problems. Likely a general theorem could be established for this class of problems, that states that the dynamic programming solutions to all combinatorial optimization problems of a certain type yield a greedy solution in a certain limit. One necessary limitation to such extensions is that in order for an integral approximation to be possible, the number of constraints cannot scale linearly with N [SM95], and thus problems like the Traveling Salesperson Problem would be resistant to this approach.

In all, this work represents a new way to frame the value of statistical physics to combinatorial optimization. For many decades, computational statistical physics algorithms have been used to stochastically search for the solutions to combinatorial optimization problems. And combinatorial optimization problems themselves have been used to study the properties of rugged landscapes in statistical physics systems. In contrast to these previous approaches, the main value of the current work is not a new highly performant algorithm, or a new context in which to apply the replica-method, but a potentially new way to understand the relationships between solutions to combinatorial optimization problems with constraints. Greedy algorithms are known ways to approximate the solutions to combinatorial optimization problems. These approaches are understood as distinct from dynamic programming which are grounded in recursive solutions to a problem. Here we have shown how each can be represented as a branch from the common starting point of the statistical physics formulation of the KP.

8 Conclusions

We represent the Knapsack Problem (KP) as a statistical physics system and use the representation to obtain three algorithms for the KP. The relationships between the algorithms reveal relationships between well-known algorithms for the KP. In particular, taking a large W limit of a dynamic programming solution to the KP yields a greedy solution to the KP.

9 Acknowledgements

This work was completed in Jellyfish. The author thanks Nicholas Arcolano, Arlo Clarke, Amir Bitran, Rostam Razban, Michael Brenner, and Pankaj Mehta for useful comments.

10 Funding

This research received no specific grant from any funding agency, commercial, or nonprofit organization.

11 Conflicts of Interest

The author has no conflicts of interest to disclose.

12 Data Availability Statement

The numerical work was done with a 2.2 GHz 6-Core Intel Core i7 processor using Python 3.9 to computationally implement all the algorithms. Code used to generate all figures is found in the repository https://github.com/mowillia/LargeWKP.

A Dynamic Programming KP from Partition Function

In this section, we show that the standard dynamic programming solution to the KP is contained within the exact KP partition function. We recall that the KP partition function

$$Z_N(\beta\mathbf{v}, \mathbf{w}, W) = \sum_{\mathbf{x}} \Theta\Big(W - \mathbf{w}\cdot\mathbf{x}\Big) \exp\Big(\beta\mathbf{v}\cdot\mathbf{x}\Big), \tag{48}$$

has the associated identity

$$Z_N(\beta\mathbf{v}, \mathbf{w}, W) = Z_{N-1}^{(N)}(\beta\mathbf{v}, \mathbf{w}, W) + e^{\beta v_N} Z_{N-1}^{(N)}(\beta\mathbf{v}, \mathbf{w}, W - w_N), \tag{49}$$

where $Z_{N-1}^{(N)}(\beta\mathbf{v}, \mathbf{w}, W)$ is the partition function in which the Nth component is eliminated from both $\mathbf{v}$ and $\mathbf{w}$, and thus only $N-1$ items are under consideration.

Using Eq.(48), the average value $\langle V\rangle_{N,W} \equiv \langle \mathbf{v}\cdot\mathbf{x}\rangle_{N,W}$ can be written in terms of the β derivative of the partition function:

$$\langle V\rangle_{N,W} = \frac{1}{Z_N(\beta\mathbf{v},\mathbf{w},W)}\frac{\partial}{\partial\beta}Z_N(\beta\mathbf{v},\mathbf{w},W) \tag{50}$$

where subscripts N, W signify that the average is defined for a weight limit W with N items under consideration. Differentiating Eq.(49) with respect to β, dividing the result by $Z_N(\beta\mathbf{v},\mathbf{w},W)$, and using Eq.(50) then yields

$$\langle V\rangle_{N,W} = \frac{\partial_\beta Z^{(N)}_{N-1}(\beta\mathbf{v},\mathbf{w},W) + e^{\beta v_N}(v_N+\partial_\beta)Z^{(N)}_{N-1}(\beta\mathbf{v},\mathbf{w},W-w_N)}{Z^{(N)}_{N-1}(\beta\mathbf{v},\mathbf{w},W) + e^{\beta v_N}Z^{(N)}_{N-1}(\beta\mathbf{v},\mathbf{w},W-w_N)} \tag{51}$$

We know that given the definition of $\langle V\rangle_{N,W}$ as a statistical physics average and $V_N(W)$ as the maximum total-value of the corresponding optimization problem, we have the equality

$$V_N(W) = \lim_{\beta\to\infty}\langle V\rangle_{N,W}. \tag{52}$$

And so, taking the limit of Eq.(51) as $\beta\to\infty$, we obtain

$$V_N(W) = \lim_{\beta\to\infty}\frac{\langle V\rangle_{N-1,W} + \lambda_{N,W}(v_N + \langle V\rangle_{N-1,W-w_N})}{1+\lambda_{N,W}} \tag{53}$$

where we used Eq.(50) in the final equality, and we defined

$$\lambda_{N,W} \equiv \frac{e^{\beta v_N}Z^{(N)}_{N-1}(\beta\mathbf{v},\mathbf{w},W-w_N)}{Z^{(N)}_{N-1}(\beta\mathbf{v},\mathbf{w},W)}. \tag{54}$$

In order to compute the right hand side of Eq.(53), we first note that there are three possible cases in which the limit can be evaluated: The case where $W < w_N$; the case where $W \geq w_N$ with the numerator of Eq.(54) dominating the denominator for $\beta\to\infty$; the case where $W \geq w_N$ with the denominator of Eq.(54) dominating the numerator for $\beta\to\infty$.

For the case where $W < w_N$, the second term in Eq.(49) (or, equivalently, the second term in the numerator or denominator of Eq.(53)) vanishes because the partition function Z_N cannot be defined for negative weight. Thus, we find $\lambda_{N,W} = 0$ for $W < w_N$, and applying the definition Eq.(52) to the right hand side of Eq.(53), we find

$$V_N(W) = V_{N-1}(W) \qquad [\text{for } W < w_N]. \tag{55}$$

For the next two cases defined by $W \geq w$, we recall that the partition function is a Boltzmann weighted sum over states where the argument of the Boltzmann weight is proportional to the total value of the collection for the corresponding state. When we take $\beta\to\infty$, the differences in the total values between states become more pronounced so that the partition function effectively becomes defined by its maximum total-value state. Specifically, in this limit, a general partition function Z becomes $Z = e^{\beta\mathcal{O}_{\max}} + \cdots$ where $\mathcal{O}_{\max}$ is the maximum total-value and "$\cdots$" stands for sub-leading terms in this limit.

Thus whether the numerator of Eq.(54) dominates the denominator in the $\beta \to \infty$ limit is entirely dependent on the relative values of the maximum total-values associated with each partition function: If the maximum total-value for $Z^{(N)}_{N-1}(\beta\mathbf{v}, \mathbf{w}, W)$ is greater than that for $e^{\beta v_N} Z^{(N)}_{N-1}(\beta\mathbf{v}, \mathbf{w}, W - w_N)$ then $\lim_{\beta\to\infty} \lambda_{N,W} = 0$. Alternatively, if the maximum total-value for $Z^{(N)}_{N-1}(\beta\mathbf{v}, \mathbf{w}, W)$ is less than that for $e^{\beta v_N} Z^{(N)}_{N-1}(\beta\mathbf{v}, \mathbf{w}, W - w_N)$ then $\lim_{\beta\to\infty} \lambda_{N,W} = \infty$. For the partition function $Z^{(N)}_{N-1}(\beta\mathbf{v}, \mathbf{w}, W)$, the maximum total-value is by definition $V_{N-1}(W)$, and for the partition function $e^{\beta v_N} Z^{(N)}_{N-1}(\beta\mathbf{v}, \mathbf{w}, W - w_N)$, the maximum total-value is $v_N + V_{N-1}(W - w_N)$ where we have v_N as a first term since the factor $e^{\beta v_N}$ adds a value-element of v_N to each state.

Having determined the maximum total-values associated with the numerator and denominator of Eq.(54), and given that the relative values of these maxima determine whether $\lambda_{N,W} \to 0$ or $\to \infty$ in the $\beta \to \infty$ limit, we find that Eq.(53) (for $W \geq w_N$) becomes

$$V_N(W) = V_{N-1}(W) \qquad [\text{for } V_{N-1}(W) > v_N + V_{N-1}(W - w_N)], \tag{56}$$

and the opposite case (still with $W \geq w_N$) yields

$$V_N(W) = v_N + V_{N-1}(W - w_N) \qquad [\text{for } V_{N-1}(W) < v_N + V_{N-1}(W - w_N)]. \tag{57}$$

Reviewing the cases Eq.(55), Eq.(56), and Eq.(57), we see that together they produce the solution

$$V_N(W) = \begin{cases} V_{N-1}(W) & \text{for } W < w_N \\ \max\{V_{N-1}(W),\, v_N + V_{N-1}(W - w_N)\} & \text{for } W \geq w_N, \end{cases} \tag{58}$$

where $V_j(Q)$ is the maximum total-value of the knapsack with the first j items included and a weight-limit Q. Eq.(58) is the standard dynamic programming solution to the KP and we can thus conclude that the KP partition function reproduces the standard dynamic programming solution to the KP.

B Variations

In this appendix, we discuss the various generalizations of the 0-1 KP: The bounded, unbounded, and continuous [KPP04a] KPs. All generalizations keep the essential problem format of Eq.(1) except each changes the state space available to each object and consequently also changes the associated sum over states. This change in summation alters the solubility of the final partition function and our ability to approximate it. We find that although we can develop algorithms analogous to those found in the main text for the bounded and unbounded KPs, we can only write down the partition function for the continuous KP.

B.1 Bounded Knapsack Problem

Here we sketch the large W algorithm for the bounded version of the KP. We focus on writing the results since the relevant derivations are largely identical to those for the 0-1 problem.

B.1 Bounded Knapsack Problem

In the bounded KP, we allow each object i to appear a maximum of C_i times in the final collection. We collectively represent these maximum constraints through the vector $\mathbf{C} = (C_1, C_2, \ldots, C_N)$. For the statistical-physics representation of the problem, this change amounts to replacing the summation Eq.(4) with

$$\sum_{\mathbf{x}} \equiv \prod_{j=1}^{N} \sum_{x_j=0}^{C_j} \qquad \text{[Summation for "multiple copies" problem]}, \tag{59}$$

where x_i denotes the number of times object i is included in the final collection. Following a derivation similar to that in Sec. 2, we find that the partition function for this system is

$$Z_N\left(\beta\mathbf{v}, \mathbf{w}, \mathbf{C}, W\right) = \frac{1}{2\pi i}\oint_{\Gamma} \frac{dz}{z^{W+1}}\frac{1}{1-z}\prod_{k=1}^{N}\frac{1 - z^{(C_k+1)w_k}e^{(C_k+1)\beta v_k}}{1 - z^{w_k}e^{\beta v_k}}. \tag{60}$$

For this partition function, we can derive a recursive relation analogous to Eq.(11) and ultimately connect Eq.(60) to the dynamic programming solution to the bounded KP. Noting that

$$\frac{1 - z^{(C_N+1)w_N}e^{(C_N+1)\beta v_N}}{1 - z^{w_N}e^{\beta v_N}} = 1 + z^{w_N}e^{\beta v_N}\frac{1 - z^{C_N w_W}e^{C_N\beta v_N}}{1 - z^{w_N}e^{\beta v_N}} \tag{61}$$

we find that Eq.(60) implies

$$Z_N\left(\beta\mathbf{v}, \mathbf{w}, \mathbf{C}, W\right) = Z_{N-1}^{(N)}\left(\beta\mathbf{v}, \mathbf{w}, \mathbf{C}, W\right) + e^{\beta v_N} Z_N\big(\beta\mathbf{v}, \mathbf{w}, \mathbf{C}^{(N)}, W - w_N\big) \tag{62}$$

where $Z_{N-1}^{(k)}\left(\beta\mathbf{v}, \mathbf{w}, \mathbf{C}, W\right)$ is the partition function Eq.(60) where the kth component is eliminated from all the vectors and thus only $N-1$ components are under consideration, and we defined $\mathbf{C}^{(k)} \equiv (C_1, \ldots, C_k - 1, \ldots, C_N)$. Letting $V_k(W, \mathbf{C})$ be the optimal value for the bounded KP instance where only the first $k \leq N$ items are used, we can show (using arguments similar to those Appendix A) that Eq.(62) implies

$$V_N(W, \mathbf{C}) = \begin{cases} V_{N-1}(W, \mathbf{C}) & \text{for } W < w_N \\ \max\{V_{N-1}(W, \mathbf{C}),\, v_N + V_N(W - w_N, \mathbf{C}^{(N)})\} & \text{for } W \geq w_N \end{cases}, \tag{63}$$

which defines the dynamic programming algorithm for the bounded KP.

With Eq.(60), we can also apply the method of steepest descent (as in Sec. 3) to approximate this partition function and derive the expressions needed to formulate algorithms akin to those in Sec. 4.

For these new algorithms, we primarily need new expressions for Eq.(27), Eq.(28), and Eq.(41). Deriving an expression analogous to Eq.(27) is straightforward; the calculation is identical to that for the 0-1 problem. We find

$$G_N\left(z; \beta\mathbf{v}, \mathbf{w}, \mathbf{C}, W\right) = -W + \frac{z_0}{1-z_0} - \sum_{i=1}^{N}\frac{w_i}{1 - e^{-v_i/T}z_0^{-w_i}} + \sum_{i=1}^{N}\frac{(C_i+1)w_i}{1 - e^{-(C_i+1)v_i/T}z_0^{-(C_i+1)w_i}}. \tag{64}$$

Similarly, the expression for the analog of Eq.(28) can be found by differentiating the potential term inferred from Eq.(60). From this differentiation we find

$$\langle x_\ell \rangle = \frac{C_\ell + 1}{1 - e^{-(C_\ell+1)v_\ell/T} z_0^{-(C_\ell+1)w_\ell}} - \frac{1}{1 - e^{-v_\ell/T} z_0^{-w_\ell}} + \mathcal{O}(w_j/W). \tag{65}$$

From here, our formulation of the algorithm can proceed in one of two directions to obtain X_ℓ: We can take the $T \to 0$ limit of Eq.(65); or we can consider Eq.(65) with a finite threshold that allows us to determine the assignment of objects.

Following the first path, we consider the function

$$f_B(x;\alpha) = \frac{B+1}{1 - e^{-(B+1)x/\alpha}} - \frac{1}{1 - e^{-x/\alpha}}. \tag{66}$$

Combining each term under a common denominator and applying L'Hopital's rule twice yields $f_B(0;\alpha) = B/2$. Taking $\alpha \to 0$, we then find $\lim_{\alpha\to 0} f_B(x;\alpha) = BH(x)$ where $H(x)$ is the Heaviside step function that is defined at zero by $H(0) = 1/2$. Therefore, we see that the $T \to 0$ limit of Eq.(65) is

$$X_\ell = \lim_{T\to 0} \langle x_\ell \rangle = C_\ell H(v_\ell - \gamma_0 w_\ell) + \mathcal{O}(w_j/W).,\tag{67}$$

where $\gamma_0 = \lim_{T\to 0} \gamma(T)$, $\gamma(T) \equiv T \ln z_0(T)$, and $z_0(T)$ is the solution obtained from the constraint condition Eq.(64) being set to zero. Eq.(67) allows us to formulate the $T = 0$ algorithm for the bounded KP.

For the second path, we have to use a threshold that can convert $\langle x_\ell \rangle$ to an integer. We introduce this threshold so that we can round $\langle x_\ell \rangle$ to the nearest integer based on how stringent we want to be about the rounding. To this end, we will define a $p_{\text{thresh}} \in (0,1)$ and define the solution X_ℓ as

$$X_\ell = \begin{cases} C_\ell & \text{if } \langle x_\ell \rangle > C_\ell\, p_{\text{thresh}} \\ 0 & \text{otherwise.} \end{cases} \tag{68}$$

In Eq.(68), we have forced the occupancy for object ℓ to either be zero or its maximum number of instances in the collection. We chose this ”all or nothing” decision criterion in order to have a solution similar to that for Eq.(29). Such a choice also well corresponds with the $T = 0$ solution Eq.(67).

B.2 Unbounded Knapsack Problem

Here we sketch the large W algorithm for the unbounded version of the KP. We again focus on writing the results since the relevant derivations are largely identical to those for the 0-1 problem.

In the unbounded KP, we allow each object i to appear a countably infinite number of times in the final collection. For our statistical-physics representation of the problem this change amounts to replacing the summation Eq.(4) with

$$\sum_{\mathbf{x}} \equiv \prod_{j=1}^{N} \sum_{x_j=0}^{\infty} \qquad \text{[Summation for "infinite" problem]}, \tag{69}$$

where x_i denotes the number of times object i is included in the final collection. If we were to follow a derivation similar to that in Sec. 2, we would reach the expression

$$Z_{N,\infty}(\beta\mathbf{v}, \mathbf{w}, W) = \frac{1}{2\pi i} \oint_{\Gamma} \frac{dz}{z^{W+1}} \frac{1}{1-z} \prod_{k=1}^{N} \sum_{x_k=0}^{\infty} \left(z^{w_k} e^{\beta v_k}\right)^{x_k}. \tag{70}$$

This expression contains an infinite series, and for such series there are conditions on whether the final result is finite. In this case, in order for Eq.(70) to be finite we require

$$|z| < e^{-\beta v_k / w_k} \quad \text{for } k = 1, 2, \ldots, N. \tag{71}$$

We can ensure this condition by choosing a contour Γ_R whose z values satisfy $z \in R$ where R is the circle in the complex plane with radius equal to $\min(e^{-\beta v_1/w_1}, e^{-\beta v_2/w_2}, \ldots, e^{-\beta v_N/w_N})$.

Constraining our contour in this way, we find that the partition function is

$$Z_{N,\infty}(\beta\mathbf{v}, \mathbf{w}, W) = \frac{1}{2\pi i} \oint_{\Gamma_R} \frac{dz}{z} \exp F_{N,\infty}(z; \beta\mathbf{v}, \mathbf{w}, W), \tag{72}$$

where we defined

$$F_{N,\infty}(z; \beta\mathbf{v}, \mathbf{w}, W) \equiv -W \ln z - \ln(1-z) - \sum_{k=1}^{N} \ln\left(1 - z^{w_k} e^{\beta v_k}\right) \tag{73}$$

With Eq.(72), we can apply the method of steepest descent as in Sec. 3 to approximate this partition function and derive the necessary expressions to formulate algorithms akin to those for the 0-1 problem in Sec. 4. Namely, we can derive expressions analogous to Eq.(27) and Eq.(28), or Eq.(64) and Eq.(65)

However, the z value at which $F_{N,\infty}$ is minimized is not always included within the region of contours for which the infinite series that takes us from Eq.(70) to Eq.(72) is valid. In other words, the z that minimizes Eq.(73) can violate Eq.(71) and thus exist off the contour Γ_R. In such a case, we cannot sensibly evaluate the integrand of the partition function at this minimum potential value. Therefore, the method of steepest descent cannot always be applied to the partition function in the unbounded case.

We can rectify this by noting that no non-trivial KP is truly unbounded: If each item has a weight, there must be a maximum number of each item that can be included in the knapsack. Thus we can convert the unbounded KP into a bounded KP where the bound is defined by

$$C_\ell \equiv \lfloor W/w_\ell \rfloor. \tag{74}$$

Thus, the large W algorithms for the unbounded case are identical to the algorithms sketched in Appendix B.1 except that the components of **C**, rather than being a set of independently defined problem parameters, are computed from the weights and weight-limit through Eq.(74).

B.3 Continuous Knapsack Problem

Unlike the bounded and unbounded KPs, the continuous KP has a continuous space of states. Consequently, the previous discrete delta and Heaviside function methods do not apply, and we cannot reach an expression to which we can apply the method of steepest descent. Therefore, it seems that we cannot obtain a statistical-physics-based algorithm for this case. Still, it is possible to achieve the limited victory of writing a partition function for this problem and we do so here.

In the continuous KP, x_i can take on any real number between 0 and 1 inclusive, and each x_i represents the amount of object i we include in the final collection. We take $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ to represent the amounts of each object in the collection, and the total weight and total value of the objects to be $\mathbf{w} \cdot \mathbf{x}$ and $\mathbf{v} \cdot \mathbf{x}$, respectively, where $\mathbf{w} = (w_1, w_2, \ldots, w_N)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_N)$ are the corresponding weight and value vectors.

Generalizing our previous discrete-space analysis to this continuous-space case amounts to a change from a discrete sum to a continuous integration. The partition function for this continuous case is

$$Z_N^{\text{cont}}(\beta\mathbf{v}, \mathbf{w}, W) = \int_0^1 d^N\mathbf{x}\, \overline{\Theta}\,(W - \mathbf{w}\cdot\mathbf{x}\,)\exp(\,\beta\mathbf{v}\cdot\mathbf{x}\,)\,, \tag{75}$$

where

$$\int_0^1 d^N\mathbf{x} \equiv \int_0^1 dx_1 \int_0^1 dx_2 \ldots \int_0^1 dx_N \tag{76}$$

is the integration over the state space and where

$$\overline{\Theta}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise,} \end{cases} \tag{77}$$

is the Heaviside step function with a continuous argument. With such a continuous argument, the step function has the integral representation

$$\overline{\Theta}(x) = \frac{1}{2\pi i}\lim_{\varepsilon\to 0^+}\int_{-\infty}^{\infty} dk\, \frac{e^{ikx}}{k - i\varepsilon}, \tag{78}$$

which can be affirmed by transforming the real-space integral in k into a closed contour integral in the upper-half of the complex plane. Inserting this expression into Eq.(75), we obtain

$$\begin{aligned} Z_N^{\text{cont}}(\beta\mathbf{v}, \mathbf{w}, W) &= \int_0^1 d^N\mathbf{x}\, \frac{1}{2\pi i}\lim_{\varepsilon\to 0^+}\int_{-\infty}^{\infty} dk\, \frac{e^{ik(W-\mathbf{w}\cdot\mathbf{x})+\beta\mathbf{v}\cdot\mathbf{x}}}{k - i\varepsilon} \\ &= \frac{1}{2\pi i}\lim_{\varepsilon\to 0^+}\int_{-\infty}^{\infty} dk\, \frac{e^{ikW}}{k - i\varepsilon}\int_0^1 d^N\mathbf{x}\, e^{\mathbf{x}\cdot(\beta\mathbf{v} - ik\mathbf{w})} \end{aligned}$$

$$= \frac{1}{2\pi i} \lim_{\varepsilon \to 0^+} \int_{-\infty}^{\infty} dk \, \frac{e^{ikW}}{k - i\varepsilon} \prod_{j=1}^{N} \int_{0}^{1} dx_j \, e^{x_j(\beta v_j - ikw_j)}, \tag{79}$$

ultimately yielding the partition function

$$Z_N^{\text{cont}}(\beta \mathbf{v}, \mathbf{w}, W) = \frac{1}{2\pi i} \lim_{\varepsilon \to 0^+} \int_{-\infty}^{\infty} dk \, \frac{e^{ikW}}{k - i\varepsilon} \prod_{j=1}^{N} \frac{e^{(\beta v_j - ikw_j)} - 1}{\beta v_j - ikw_j}. \tag{80}$$

Since we are no longer integrating over a contour in the complex plane, applying a large W approximation scheme to Eq.(80) would require a slightly different formalism from that presented in this paper. Interestingly, fast approximation schemes are not needed for the continuous KP as it can be solved exactly via an $O(N \log N)$ greedy algorithm [Dan57] and thus what would typically be the fastest "approximation" method actually yields the exact solution.

C Additional Instances

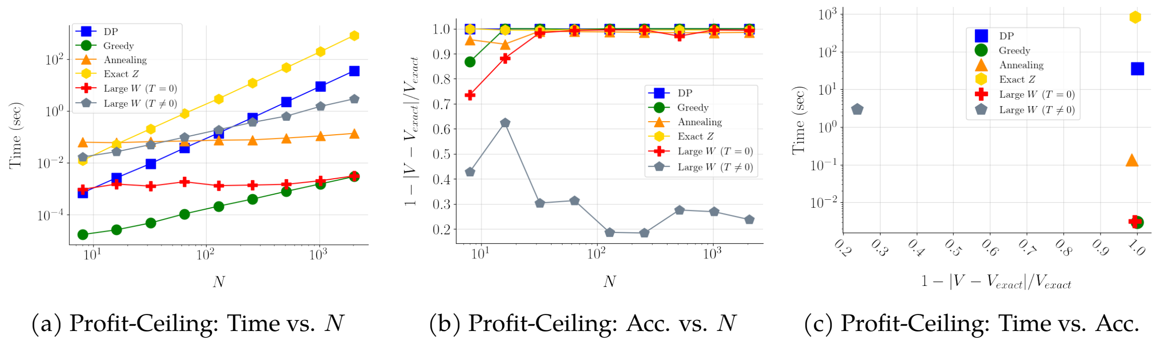
We consider the runtime and accuracy results for the two other special instances introduced in [Pis05]: The "profit-ceiling instance" and the "multiple strongly correlated Items" instance.

- **Profit-Ceiling Instances:** Values are multiples of a given integer parameter d. The instance is generated by randomly (and uniformly) selecting weights w in the interval $[1, R]$, for a free integer parameter R, and then setting the values to $p_j = d\lceil w_j/d \rceil$. [Pis05] noted that particularly difficult instances appeared by choosing $d = 3$. The value of R was set to 100.

- **Multiple Strongly Correlated Items Instances:** Values are related to weights through $v = w + k_i$ where k_i for $i = 1, 2$ for the case of two strongly correlated instances. Specifically, the instances are generated as follows: The weights of N items are randomly distributed in $[1, R]$. If the weight w_j is divisible by a chosen d, we set the value $v_j = w_j + k_1$, otherwise we set it to $v_j = w_j + k_2$.

  In accordance with [Pis05], we set $d = 6$, $k_1 = 3R/10$, and $k_2 = 2R/10$. The value of R was set to 100.

The results of applying the three introduced algorithms and the three standard algorithms to these instances are shown in in Fig. 5. The results are similar to those in Fig. 3, except since neither the "multi-strong" or the "profit-ceiling" instances exhibit high degeneracies in their value-to-weight ratios, we find that the zero-temperature algorithm performs better than it did for the "spanner" instance. For both types of instances, the standard greedy algorithm achieves highly accurate results as N increases (due to the $W \gg w_j$ limit), and the zero-temperature algorithm improves as N increases but remains bounded above by the standard greedy algorithm. The nonzero-temperature algorithm fails to achieve accurate results for the profit-ceiling instance, but maintains an accuracy above 90% for the multiple-strongly correlated items instance. Fig. 5c and Fig. 5f show the close proximity in time and accuracy space between the zero-temperature algorithm and the standard greedy algorithm which further motivates identifying the former as a kind of greedy algorithm. In all, these results affirm those in the text and provide additional instances where the


(a) Profit-Ceiling: Time vs. N (b) Profit-Ceiling: Acc. vs. N (c) Profit-Ceiling: Time vs. Acc.

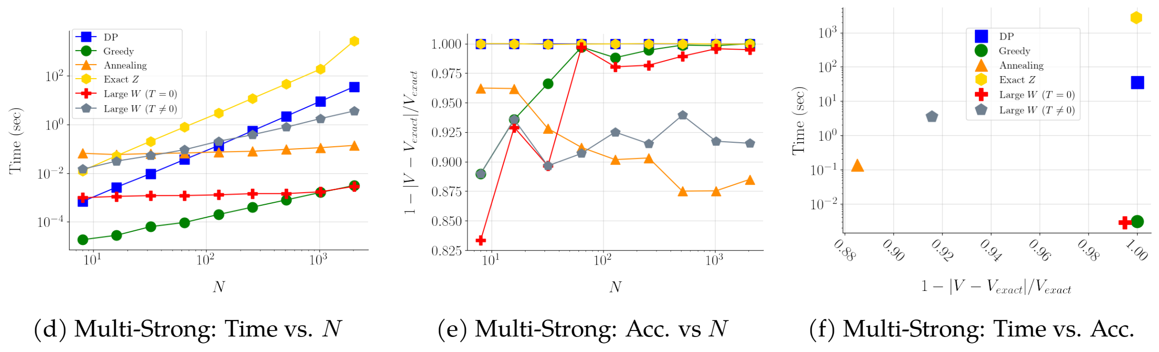

(d) Multi-Strong: Time vs. N (e) Multi-Strong: Acc. vs N (f) Multi-Strong: Time vs. Acc.

Figure 5: Runtime and Accuracy Comparisons: In (a),(b), and (c), we show the results for the ”profit-ceiling instances” and in (d), (e), and (f), we show the results for the ”multiple strongly correlated items instances.” (a) and (d) depict the runtimes as a function of N. (b) and (e) depict the accuracy of the algorithms as a function of N. (c) and (f) depict the runtime and accuracy for the instance with $N = 2048$ items. For the temperature dependent algorithms we set $T = 1.0$. The zero-temperature algorithm has increasing performance with increasing N as is consistent with Fig. 1. Code and results used to generate these figures is linked to in Sec. 12, *Data Availability Statement*.

introduced algorithms do not perform better in runtime or accuracy than their standard counterparts.

References

[And86] Philip W Anderson. What statistical mechanics has to say to computer scientists. *Physica A: Statistical Mechanics and its Applications*, 140(1-2):405–409, 1986.

[And88] C Perterson J Anderson. Neural networks and np-complete optimization problems; a performance study on the graph bisection problem. *Complex Systems*, 2:58–59, 1988.

[Bax16] Rodney J Baxter. *Exactly solved models in statistical mechanics*. Elsevier, 2016.

[Čer85] Vladimír Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[Com21] SciPy Community. scipy.optimize.fsolve. *SciPy v1.6.1 Reference Guide*, Mar 2021. URL: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html`.

[Dan57] George B Dantzig. Discrete-variable extremum problems. *Operations research*, 5(2):266–288, 1957.

[Eli43] TS Eliot. Little gidding. four quartets. *Selected poems*, 1943.

[Fon95] José Fernando Fontanari. A statistical analysis of the knapsack problem. *Journal of Physics A: Mathematical and General*, 28(17):4751, 1995.

[Has13] Sadri Hassani. *Mathematical physics: a modern introduction to its foundations*, chapter Method of Steepest Descent. Number 11.5. Springer Science & Business Media, 2013.

[Ing93] Lester Ingber. Simulated annealing: Practice versus theory. *Mathematical and computer modelling*, 18(11):29–57, 1993.

[Ino97] Jun-ichi Inoue. Statistical mechanics of the multi-constraint continuous knapsack problem. *Journal of Physics A: Mathematical and General*, 30(4):1047, 1997.

[ISTO07] Shotaro Inoue, Shigeru Sugiyama, Andrew A Travers, and Takashi Ohyama. Self-assembly of double-stranded dna molecules at nanomolar concentrations. *Biochemistry*, 46(1):164–171, 2007.

[KOL94] E Korutcheva, M Opper, and B Lopez. Statistical mechanics of the knapsack problem. *Journal of Physics A: Mathematical and General*, 27(18):L645, 1994.

[KPP04a] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*. Springer, 2004.

[KPP04b] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems*, chapter Dynamic Programming, pages 20–27. Number 2.3. Springer, 2004.

[KPP04c] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems*, chapter Greedy Algorithm, pages 15–16. Number 2.1. Springer, 2004.

[McQ73] Donald McQuarrie. *Statistical Mechanics*. Harper and Row, 1973.

[MM09] Marc Mezard and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

[Nis01] Hidetoshi Nishimori. *Statistical physics of spin glasses and information processing: an introduction*, volume 111. Clarendon Press, 2001.

[PIM06] Allon Percus, Gabriel Istrate, and Cristopher Moore. *Computational complexity and statistical physics*. Oxford University Press, USA, 2006.

[Pis05] David Pisinger. Where are the hard knapsack problems? *Computers & Operations Research*, 32(9):2271–2284, 2005.

[R$^+$64] Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-Hill New York, 1964.

[Res91] Wolfram Research. Nsolve. *Wolfram Language function*, 1991. URL: https://reference.wolfram.com/language/ref/NSolve.html.

[ros19] Knapsack problem/0-1. *Rosetta Code*, Sept 2019. URL: https://web.archive.org/web/20190908102907/http://www.rosettacode.org/wiki/Knapsack_problem/0-1 [cited 2020].

[SKL$^+$13] Lloyd M Smith, Neil L Kelleher, Michal Linial, David Goodlett, Pat Langridge-Smith, Young Ah Goo, George Safford, Leo Bonilla, George Kruppa, Roman Zubarev, et al. Proteoform: a single term describing protein complexity. *Nature methods*, 10(3):186, 2013.

[SM95] Zhenming Shun and Peter McCullagh. Laplace approximation of high dimensional integrals. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(4):749–760, 1995.

[TM90] Paolo Toth and Silvano Martello. *Knapsack problems: Algorithms and computer implementations*. Wiley, 1990.