



Deposited via The University of York.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/id/eprint/241626/>

Preprint:

Park, Jessica, Stepney, Susan and D'Amico, Irene (2026) Benchmarking the ORCA PT-2 Boson Sampler using Minimum Dominating Set Problems. [Preprint]

<https://doi.org/10.48550/arXiv.2605.30935>

Reuse

Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Benchmarking the ORCA PT-2 Boson Sampler using Minimum Dominating Set Problems

Jessica Park^{1,2*}, Susan Stepney^{1†} and Irene D’Amico^{2†}

^{1*}Department of Computer Science, University of York, York, UK.

²Department of Physics, University of York, York, UK.

*Corresponding author(s). E-mail(s): jpg567@york.ac.uk;

Contributing authors: susan.stepney@york.ac.uk;

irene.damico@york.ac.uk;

[†]These authors contributed equally to this work.

Abstract

We use boson sampling as part of a gradient-free variational algorithm (the Binary Bosonic Solver) to solve a minimum dominating set problem and compare these results to a number of exact and heuristic classical algorithms. The boson sampling has been performed on the physical PT-2 time-bin interferometer from ORCA Computing. The PT-2 device has been tested here using both a single- and double-loop configuration and the results are compared based on the best found solution and the overall run time. With the parameters used in this experiment, the boson sampler is outperformed by the classical methods, but we hypothesise that this is due to insufficient samples and iterations. We classically simulate boson sampling in a single-loop configuration to break down the run-time for individual algorithmic components, allowing for estimates of when boson sampling may outperform classical methods. This study recommends a watching brief on boson sampling as the complexity of the interferometer is improved and the loss in the hardware is reduced allowing for better performance from the associated algorithms.

Keywords: Optimisation, Boson Sampling, Benchmarking

1 Introduction

Boson sampling is a quantum computing paradigm that is based on determining the output distribution from bosons in linear interferometers based on knowledge of the beam splitter locations and angles, the number of input bosons, and the number of output modes. Classically, this is computationally very expensive to calculate ($\#P$ complete, so at least as hard as NP-complete) with an exponential number of terms. However, a boson sampling device enacts the chosen interferometer and reports the output (Aaronson & Arkhipov, 2011). Although a boson sampler can produce this output distribution more effectively than a classical computer could calculate it, there is ongoing debate as to whether a boson sampler can solve any real problems with a quantum advantage (Bromley et al., 2020). It has been shown that boson samplers can be used in variational algorithms (eg. combinatorial optimisation, generative adversarial networks), calculating graph properties and proof-of-work consensus, although they have yet to overtake the utility of classical supercomputers (Arrazola & Bromley, 2018; Bacarreza et al., 2025; Slys, Kurowski, & Waligóra, 2024).

Our previous work (Park, Stepney, & D’Amico, 2026) uses simulation to show that a boson sampler can, in principle, be used to find solutions to dominating set problems that are of comparable quality to typical classical methods. That work uses the ORCA Computing Software Development Kit (SDK) with their Binary Bosonic Solver (BBS) to simulate both their PT-1 and PT-2 devices. The classical methods used for comparison were a greedy algorithm, a heuristic algorithm from the NetworkX Python package, and a linear programming approach using the PULP Python package. The wall-clock time required to run the boson sampler simulation is orders of magnitude larger than the classical approaches, but this is to be expected from the complexity of calculating a linear interferometer output distribution as described above.

We extend the results presented previously in this work, by replicating the benchmarking experiment using a physical ORCA PT-2 series boson sampler, rather than its simulator. The PT-1 is no longer in use, and therefore cannot be used for comparison. There have also been a number of significant updates in the BBS algorithm, designed to improve performance and runtime. Therefore this study reports new simulated results as well as the results from the real device, to ensure a fair comparison. The classical algorithm results from the original paper are repeated here for comparison purposes.

2 Background

2.1 Photonic boson sampling

A photonic boson sampler, such as the ORCA PT-2, performs calculations using samples from a distribution of identical photons after they have passed through a linear interferometer. Linear interferometers are formed of a number of optical components called beam splitters that can change the path of incident photons. Changing the ‘angle’ of the beam splitter (θ) changes the probabilities of transmission or reflection of the photon. Quantum effects, namely the Hong-Ou-Mandel effect, appear when multiple indistinguishable photons are coincident on a beam splitter. The Hong-Ou-Mandel

effect describes the quantum interference of indistinguishable photons at a beam splitter whereby the photons will always leave the beam splitter in the same output mode (Hong, Ou, & Mandel, 1987). They are then in a superposition of both output paths and the result is determined on measurement.

Photons are commonly used in boson samplers as they have long coherence times, do not interact strongly with the environment, and can function at room temperature. Interferometers can be constructed either in free space or with the photons travelling through optical fibres between beam splitters. Because they travel at the speed of light, the construction of the linear interferometer needs to be very precise to ensure that photons arrive at the beam splitters at the same time and are indistinguishable, allowing for the quantum effects to take place.

Linear interferometers are usually diagrammatically and physically spatially binned. This means that the input photons all enter the interferometer at the same time in different channels that are directed towards a rows of beam splitters. The width of the interferometer refers to how many channels the photons can enter the interferometer by and the depth refers to the number of rows of beam splitters through which the photons will pass. At the end of each channel in the interferometer, there are photon detectors which then measure the output distribution. The number of output bins is typically known as the number of modes of the interferometer.

2.2 Physical devices

ORCA Computing, among others, have a different approach to the interferometer that requires less physical space and fewer components. They design and build time-binned devices, in which the input photons enter the interferometer via a single channels at defined intervals, τ , and loops of fibre-optic cable are used as delay lines to cause photons to interact at the beam splitters (Bradler & Wallner, 2021). At each beam splitter, the photon can either move through to the next stage of the interferometer or be directed back into a fibre optic loop where it could then interact with the following photon mode at the same beam splitter. This means fewer beam-splitters are required, and only a single photon generation device. Similarly, this set-up only requires one photon detector, as the output modes, $|n_M\rangle$ are counted as the time steps in which photons exit the interferometer and are detected (He et al., 2017; Motes, Gilchrist, Dowling, & Rohde, 2014). The principle of time-bin interferometry is shown in Figure 1 and is mathematically equivalent to the spatial approach (Sempere-Llagostera, Patel, Walmsley, & Kolthammer, 2022).

Our previous work (Park et al., 2026) mainly considers the ORCA PT-1 device, as that was the available device at the time. The PT-2 device is mentioned in that study as a future development planned by ORCA Computing. Since then ORCA Computing have begun manufacturing, selling and operating PT-2 devices. The PT-2 device samples from non-Gaussian, photon-subtracted squeezed states, rather than single photons as in the PT-1, and uses a photon-number resolving detector (Olson, Seshadreesan, Motes, Rohde, & Dowling, 2015). There have also been engineering improvements for more stability and lower loss. The PT-2 device also has capability for running a time-bin interferometer with two loops. By analogy to the spatial interferometer, adding more loops provides additional depth or layers of beam splitters, whilst inputting more

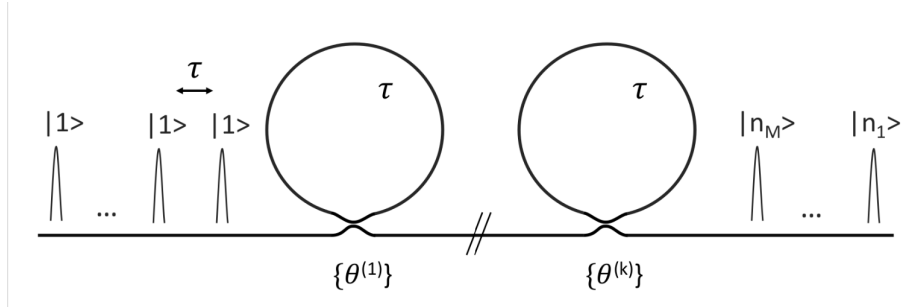


Fig. 1: Diagrammatic representation of time-bin interferometry. Taken from [Bradler and Wallner \(2021\)](#).

photons (more time-bins) is the equivalent to making the interferometer wider and having more spatial modes. Each additional loop makes the PT Series progressively harder to simulate, with only 3 loops required to reach a regime that is not simulable classically ([Deshpande et al., 2022](#); [Novák, Roberts, Makarovskiy, García-Patrón, & Clements, 2025](#)). Whether this theoretical quantum advantage translates to real world quantum utility in the kind of applications considered here remain unproven, and will depend on whether generating more complex output distributions brings an advantage in training optimisation algorithms.

2.3 Optimisation algorithm

The Binary Bosonic Solver (BBS) is a variational quantum algorithm that uses the output distribution from the time-bin interferometer to solve optimisation functions. It is an iterative approach involving the training of beam splitter parameters to optimise a given cost function. More details are available in [Makarovskiy et al. \(2025\)](#) and its application in this study is detailed in the following section.

The key development since our previous work is the option for a gradient-free version of the BBS. This requires evaluating a single observable per update step instead of multiple evaluations per update step to calculate gradients. It is expected that the repeated cost function evaluations make up a considerable portion of the total algorithm runtime; if this is so, fewer evaluations should improve runtime. The timings for the individual portions of the algorithm are discussed alongside the results below.

2.4 Application domain

Due to the significant upgrade in both the hardware and software since our original study, we do not provide a direct comparison between those results and the new simulations here. Instead, we run further simulations using the new gradient-free binary bosonic solver.

As in the original study, we are considering the use of boson sampler in the case of solving a surveillance coverage problem that can be articulated as follows. Given a map of places of interest (POIs), a graph can be drawn where each POI is a node and an edge exists between nodes where there is a line of sight between them. The surveillance

Algorithm	Hardware	Label
Binary Bosonic Solver	PT-2 (1-loop Implementation)	BBS 1-loop PT-2
Binary Bosonic Solver	PT-2 (2-loop Implementation)	BBS 2-loop PT-2
Binary Bosonic Solver	CPU (1-loop Simulation)	BBS 1-loop Sim
Greedy	CPU	Greedy
NetworkX Approximation	CPU	NX Approximation
Exact Linear Programming	CPU	PULP

Table 1: Table showing the six experiments run and compared during this study. The label column corresponds to the legends across all figures in the results section.

coverage problem is to find the smallest set of POIs that have the ability to surveil the entire map. We use here the convention that a node within the surveillance set can surveil itself. Mathematically, this is equivalent to the NP-hard *minimum dominating set* problem (Grandoni, 2006): for a graph $G = (V, E)$, find the smallest subset V' of V such that every node in the graph is adjacent to at least one node in V' .

3 Experimental Design

The majority of the experimental design follows that laid out in our earlier work (Park et al., 2026), in which the binary bosonic solver (BBS, from ORCA Computing) is applied to dominating set problems of increasing size. As discussed in the previous section, the BBS algorithm has been updated by ORCA Computing to use gradient-free optimisation. This updated version is used in this study.

These results are compared against three classical alternatives: a greedy algorithm, the NetworkX approximation algorithm, and an exact linear programming approach. These six experiments are laid out in Table 1. All of the experiments are run on identical test problems to enable accurate comparison. The CPU used to produce the classical results in this study runs an Intel i7 processor with 64GB of RAM. That methodology is summarised here with the key new developments and changes highlighted.

3.1 Dominating set cost function

The particular graphs that the algorithms are run on are the same as in the original work. They are defined as $G(n, p, s)$, where G is a graph with n nodes and a p probability of edge creation between any two nodes; s defines the random seed that is used to ensure repeatable graphs are produced. For all the results presented here we fix the graph probability density as $p = 0.05$. For meaningful results, the graph needs to be connected; this is ensured in the graph generation process. For each n , three random graphs are generated by using three different values of the seed s . This set of graphs is fixed for use with both the quantum algorithm and classical comparison.

We formulate a dominating set problem as follows. Let $\mathbf{x}$ be the bit string $\mathbf{x} = [x_0, x_1, \dots, x_{n-1}]$, where n is the number of vertices in G and $x_i = 1$ if the vertex is

in the candidate set and 0 otherwise. The neighbours of the vertex x_i are $N(x_i)$. The cost function to be minimised to find the minimum dominating set is:

$$F(\mathbf{x}) = \sum_{v_i \in V} x_i + A \sum_{v_i \in V} P_i, \quad (1)$$

where P_i is the penalty term given by

$$P_i = \begin{cases} 0, & \text{if } (x_i + \sum_{v_j \in N(v_i)} x_j) - 1 \geq 0 \\ 1, & \text{otherwise} \end{cases} \quad (2)$$

The first term in equation 1 is the size of the candidate set. The second term is the penalty term, where A is a scaling factor (in this case, $A = 2$) and P_i is a function that checks whether a vertex v_i is either part of, or neighbours a vertex in, the set. A penalty of 1 is added for every vertex in the graph that is not either part of the set or neighbours with at least one node that is. If no penalty terms are added, the set in question is a dominating set, and the result of the cost function is the size of the set. We refer to the result of the cost function, $F(x)$, as the state *energy* of that bit string.

For some problems, a minimal dominating set is not a unique solution. It is relatively easy to check whether a set of nodes dominates the graph, but without exhaustive search, it is impossible to know if it is a *minimal* dominating set. The output from each of the different algorithms is a set of nodes which is then verified to be dominating (a computationally easy step). Without knowing the true minimum set size, we are comparing the found set size from the different methods where a smaller set implies a more performant algorithm.

3.2 Optimisation algorithm pseudocode

Pseudocode of the binary bosonic solver with gradient-free optimisation is shown in Algorithm 1. The parameters $[p_i]$ (line 8) are used to define a sigmoid function that then controls the bit flipping probabilities. Line 11 is different from that used in Park et al. (2026) in that we are now using a gradient-free optimisation. The default gradient-free optimisation uses the Nevergrad package, but users can specify their own classical optimisation algorithm if desired (Teytaud & Rapin, 2018). For these tests, the default Nevergrad optimisation is used, which has a learning rate parameter λ . The other change from previous work is that we here use an all-one input state, $|1\rangle^{\otimes n}$, rather than the alternating $|01\rangle^{\otimes \frac{n}{2}}$ state from the previous study. This change was recommended to us by ORCA Computing and is considered the default in their updated software development kit.

Parameter values held constant throughout are the number of iterations (NIter = 400), the number of samples per iterations (NSamp = 600), and the learning rate used in the classically computed gradient-free optimisation ($\lambda = 1 \times 10^{-2}$).

Algorithm 1 Variational Quantum Algorithm

inputs: cost function $F(\mathbf{x})$, input state $|1\rangle^{\otimes n}$, learning rate λ

outputs: optimum state x_{min} , minimum energy E_{min}

```
1:  $\theta := [\theta_j]$ , initialise interferometer parameters randomly from  $U(0, \frac{\pi}{2})$ 
2:  $\mathbf{p} := [p_i]$ , initialise flipping parameters randomly from  $U(-5, 5)$ 
3: repeat NIter times
4:   repeat NSamp times
5:     pass input state through interferometer
6:     measure photons in each mode  $|out\rangle$ 
7:      $\mathbf{x} :=$  threshold map  $|out\rangle$  to bit string
8:      $\mathbf{x} :=$  flip or hold bits depending on  $\mathbf{p}$ 
9:     state energy  $E := F(\mathbf{x})$ 
10:   end
11:    $\theta, \mathbf{p} :=$  update with Nevergrad optimisation function  $f_{ng}(\lambda)$ 
12: end
13: return  $x_{min}, E_{min}$ 
```

3.3 Runtime components

The algorithm can be visualised as in Figure 2. This is helpful when considering the breakdown of the total runtime in terms of the individual component steps and where physically those steps are being executed.

The measures of interest in this study are the size of the dominating set found and the wall-clock runtime. The wall-clock runtime includes everything from Figure 2: the sampling from the PT device, the evaluations of the cost function on the sampler outputs, and the classically-computed updates of the interferometer parameters. The total runtime can be expressed as:

$$t_{tot} = \text{NIter}(2t_L + \text{NSamp}(t_S + t_C) + t_P) \quad (3)$$

where t_L is the classical overheads including internet latency, t_S is the time taken to produce a sample, t_C is the time per cost function evaluation and t_P is the time required to update the parameters.

In practice, the algorithm can be set to terminate when the best found solution has not changed in a given number of iterations, and we say that the training has converged. Here, however, all runs continue to the maximum number of iterations (NIter). By recording the current best solution found by each iteration we can then estimate the earliest iteration at which the algorithm could have been terminated without increasing the found set size, I_{con} . The time taken to get to this iteration, we denote as the convergence time t_{con} , which can be calculated as:

$$t_{con} = \frac{t_{tot} \times I_{con}}{\text{NIter}}, \quad (4)$$

where t_{tot} is the total runtime, and I_{con} is the iteration at which the best solution has not improved for a set number of consecutive iterations. For the results presented here, we set that number to be 50.

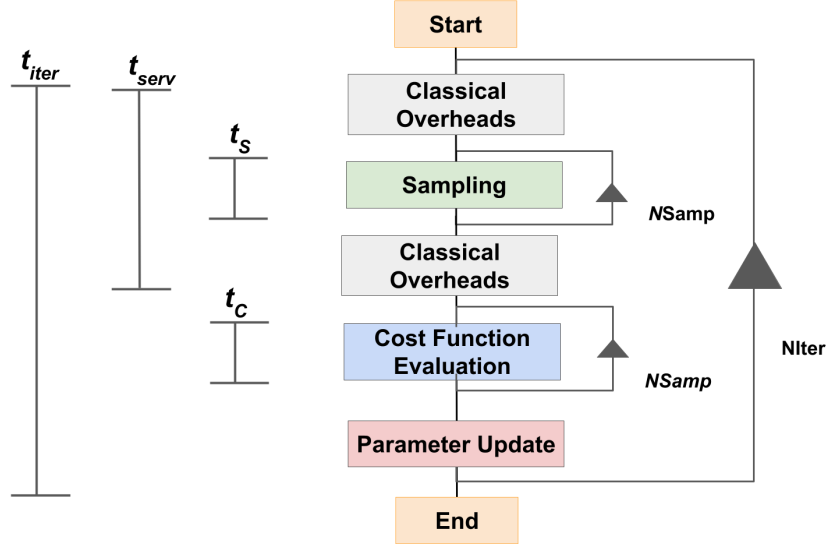


Fig. 2: A visual representation of the Binary Bosonic Solver run in practice on the PT-2 device with the various timing metrics labelled. t_{iter} is the time for a single iterations; t_{serv} is the time taken accessing the PT-2 device including classical overheads and sampling; t_s is the time for an individual sample to be generated from the PT-2 device; t_C is the time taken for the local CPU to evaluate the cost function for a single bit string instance.

As mentioned above, the PT-2 device is capable of running a time-bin interferometer with two loops. To consider the effect of the two-loop set-up, both the single and double loop configurations are run on the real hardware. Both sets of results from the ORCA PT-2 sampler (single and double loop) are compared to the classical results from the previous study.

4 Results

4.1 Solution quality: size of found sets

In terms of comparing solution quality, we look at the minimum size of the dominating set found by each method. Results come from the three classical methods (Greedy, NetworkX approximation, and PULP), and three implementations of the binary bosonic solver (BBS); a 1 loop CPU simulation (BBS 1-loop Sim), a 1-loop PT-2 implementation (BBS 1-loop PT-2), and a 2-loop PT-2 implementation (BBS 2-loop PT-2).

The results from the three classical methods are taken from the previous study (Park et al., 2026) as they are not affected by the changes to either the PT hardware or the BBS algorithm. Due to these changes, all of the BBS results (both in hardware and in simulation) are new to this paper. As discussed in the experimental design section, for each graph size, n , three random graphs are generated using three different

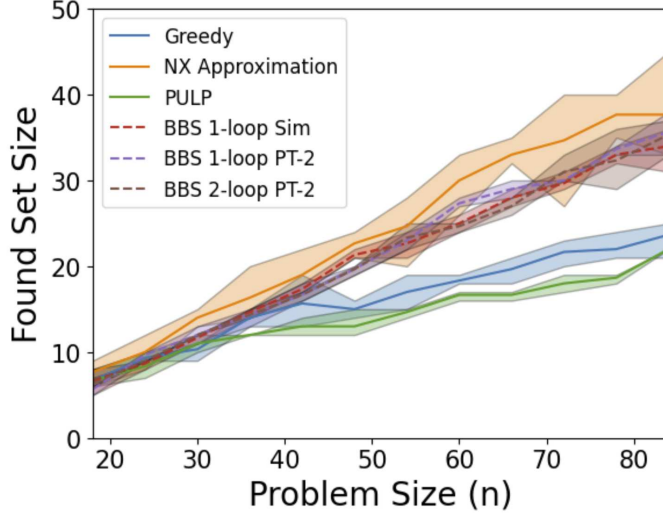


Fig. 3: Comparisons between the found solution size of three classical methods for solving the dominating set problem (Greedy, NX Approximation and PULP) and three implementations of the binary bosonic solver (BBS); a 1 loop CPU simulation (BBS 1-loop Sim), a 1-loop PT-2 implementation (BBS 1-loop PT-2), and a 2-loop PT-2 implementation (BBS 2-loop PT-2). The lines represent the mean value over three different graphs (for each value of n), and the shaded areas shows the minimum and maximum of results.

values of the seed s . These are then fixed for use with both the quantum algorithm implementations and classical comparison.

This is shown in figure 3. It has not been possible to obtain a comparable set of results for the CPU simulation of 2-loop BBS algorithm because the runtimes are increasingly prohibitive. This is discussed in more detail below.

At smaller n the BBS algorithm finds comparable set sizes to the classical algorithms in all implementations. At the larger tested n , performance of the BBS algorithm drops off. We hypothesise that this is because not enough of the search space is being explored with the chosen numbers of samples and iterations. This hypothesis could be tested in future work, but in practical applications, where use of the quantum device will likely be paid for either by runtime or by total number of samples, there may be restrictions on choice of NSamp and NIter. Being able to estimate a sufficient number of samples and iterations for a required confidence in the answer would be a beneficial study to conduct in the future.

The similarity between the BBS implemented on the 1-loop software simulator and the 1-loop PT-2 hardware helps to validate confidence in the simulation process, which may help in future work with limited access to the real device(s). There does not seem to be a discernable difference between the the performance of the one loop and two loop PT-2 implementations across the range of problem sizes tested here, even where there are smaller dominating sets available to find (as seen by the set sizes

found by the classical Greedy and PULP algorithms). The shaded areas that show the minimum and maximum sizes of the sets found are reasonably narrow, which could again suggest that the the number of samples and iterations are not adequate to search for particularly good areas of the space.

The PULP linear programming approach should provide exact solutions (and therefore a fixed benchmark) to the problem. However, in practice there are approximation errors. The problem is optimised using real numbers rather than integers, which means that when applied to an combinatorial optimisation problem such as this, some rounding is applied to the end result. This rounding can cause a suboptimal result to be output. This is seen in Figure 3 where the Greedy algorithm actually finds a smaller dominating set than PULP for one of the 30 node instances.

The results in terms of found set size are very similar to those seen in the previous study that looked only at CPU simulations: the BBS implementations find dominating sets that are smaller than the NX Approximation but larger than the other two classical methods. To effectively compare the results of this paper to the previous study it would be useful to better understand the effect of the maximum number of samples and iterations used in the BBS algorithm. These determine the number of cost function evaluations used overall and is something discussed in more detail in the remainder of this work.

4.2 Search space coverage

Unlike gradient-based optimisation methods, the gradient-free optimisation within the binary bosonic solver (BBS) algorithm does not aim to reduce the value of the cost function after every iteration, therefore this is not the best metric to track the algorithm’s progress in this case. Instead, gradient-free optimisation aims to explore the search space, and therefore it is informative to consider the number of different bit strings (candidate solutions) that are evaluated by the cost function.

The number of bit strings required to cover the entire space is 2^n ; the maximum number of different cost function evaluations in the BBS solver is $\text{NSamp} \times \text{NIter}$, which in this experiment is 240,000 (although there is no guarantee that there are no repeated bit string candidate solutions). Therefore only problems with $n \leq 17$ can be fully sampled, equivalent to exhaustive search. This can be seen in Figure 4, where the $n = 15$ line levels out at higher iterations. For $n = 18$ and $n = 21$, there is some deviation from the linear trend which suggests that there are some repeated candidate bit strings in these cases. The BBS algorithm caches previous evaluations, so the cost function is not recalculated when a previously seen bit string is generated. In all of the other, larger experiments, the number of unique cost evaluations continue to scale linearly, which implies that the BBS is succeeding in exploring new areas of the search space with each iteration.

Gradient-based optimisation is sensitive to initial parameter choice; therefore there may be a case for beginning a large optimisation problem with a gradient-free approach and later using the gradient-based version to fine-tune a close-to-optimal solution. Such a hybrid BBS approach would be an interesting and potentially beneficial avenue for future work.

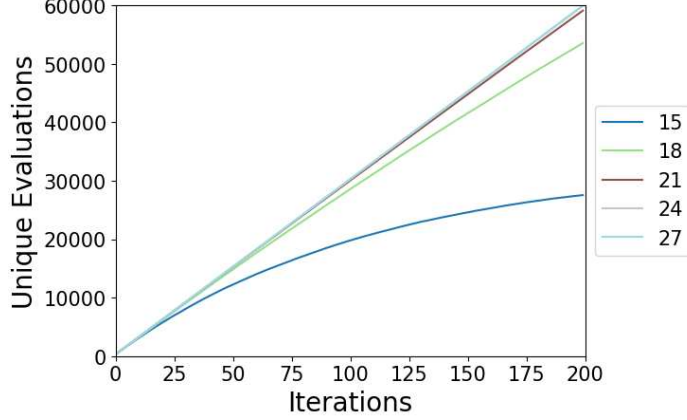


Fig. 4: The cumulative number of unique cost function evaluations carried out as the Binary Bosonic Solver progresses through the iterations, for different problem sizes.

Another benefit of the gradient-free approach is that it may find multiple distinct regions of the search space that contain near-optimal solutions. This may be particularly useful in certain use cases where a robust set of solutions is required, rather than a single result. This could allow users to down-select based on additional factors or constraints that could not be formalised in the cost function or that are unknown at the time of running the algorithm.

4.3 Run times

Figure 5 shows the wall-clock runtime for the three classical algorithms, and the three BBS implementations (1-loop Sim, 1-loop PT-2 and 2-loop PT-2). There are also lines that show the estimated convergence time as calculated by 4 for both the real and simulated PT-2 device running the gradient-free BBS algorithm. Overall the runtime for the BBS algorithm is higher than that of the classical methods whether being enacted on a CPU simulation or with the real PT-2 device. Although higher than the classical methods, the runtime of the BBS simulations in this experiment are approximately an order of magnitude smaller than shown in the previous study (Park et al., 2026). This may be due to the fewer cost function evaluations that are required per iteration of the gradient-free BBS algorithm compared to the previous gradient-based version. As in the previous study the runtime does not appear to scale with n . These further results add weight to the hypothesis formed from the previous results, that the scaling of BBS runtime is not dominated by the graph size and that this implementation could be a beneficial method for large problems if the hardware can be scaled appropriately.

It should also be noted that despite not being shown in the data collected from this experiment, the previous study showed that beyond a problem size of around $n = 100$, the PULP algorithm begins to scale exponentially in time.

The PT-2 runs (real device and simulated) include the same cost function calculations and parameter update steps run on the same CPU. This means, according

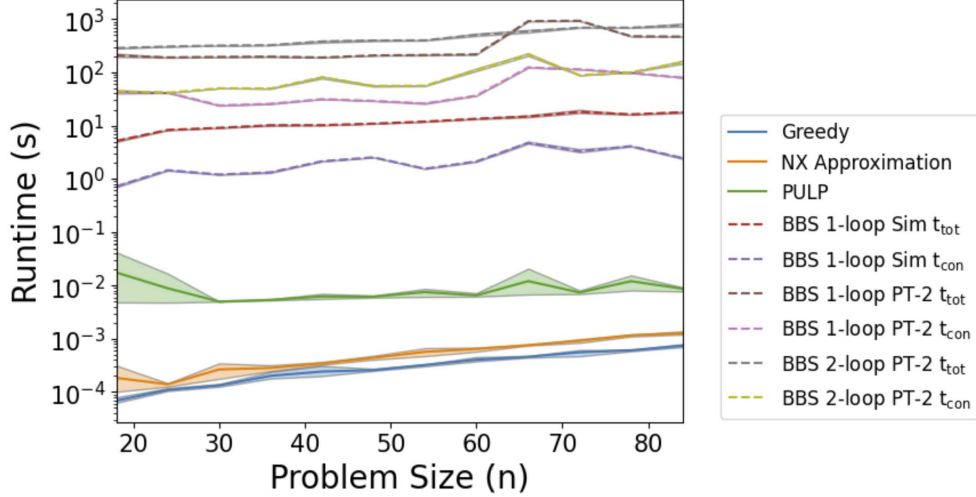


Fig. 5: Comparisons between the runtime of three classical methods for solving the dominating set problem (Greedy, NX Approximation and PULP) and three implementations of the binary bosonic solver (BBS); a 1 loop CPU simulation (BBS 1-loop Sim), a 1-loop PT-2 implementation (BBS 1-loop PT-2), and a 2-loop PT-2 implementation (BBS 2-loop PT-2). For the BBS implementations, both the total runtime t_{tot} and the estimated convergence time t_{con} are shown with convergence time, t_{con} , calculated by equation 4. Note the log scaling on the y-axis. The lines represent the mean value over three different graphs (of the same n and p values) and the shaded areas shows the minimum and maximum values of results.

to equation 3, that the difference in runtime must be due to the physical sampling. We have not been able to measure these quantities directly in this work, however ORCA Computing report that the combined classical overheads (including internet latency) and sampling time tends to be ≈ 300 ms, independent of the number of samples requested (within the tested range 1–1000). This suggests that the the actual time taken to produce the samples is orders of magnitude shorter than the classical overheads.

Figure 6 shows how time required to produce samples using the state-vector simulator of a single loop time-bin interferometer scales with both the number of samples requested and the size of the input state n (which for this study has always been equal to the size of the problem). In the CPU simulation of the boson sampler, the sampling element happens via a “chain-rule” calculation that works out the probabilities of seeing different photon numbers in the first mode of the interferometer, samples from this distribution, then uses this to calculate the conditional probability on the second mode, and so on. This calculation is repeated for every sample requested by the user. Further details of the simulation method can be found in (Bulmer et al., 2022). From figure 6, it is clear that drawing large numbers of samples linearly increases the runtime, which is expected from this method of simulation. For the range of problem sizes used in the real experiment ($n < 90$), the simulated sampling times are below

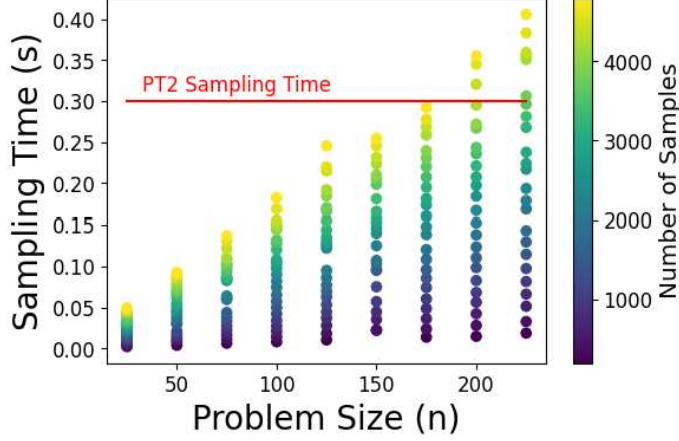


Fig. 6: Time taken to draw different numbers of samples from a simulated single loop time-bin interferometer for a range of problem sizes. The red line denotes the approximate time required to complete the equivalent step using the PT-2 device.

the combined classical overheads and sampling times quoted from ORCA for the real device (≈ 300 ms; shown as the horizontal line in Figure 6). Although the entire BBS algorithm was tested only for problem sizes $n < 90$, it was possible to measure the sampling time required for the CPU simulation for $n < 225$ and $\text{NSamp} < 5000$. Plotting these shows that, there is a point at which the PT-2 device should produce (and deliver) samples faster than the CPU simulation. This may be the point at which quantum advantage can be seen.

It is theoretically possible to simulate a full state-vector calculation of the interferometer only the once, find the complete probability distribution, then draw an unlimited number of samples at little extra cost. However, this initial calculation is what makes boson sampling a $\#P$ problem, and quickly becomes intractable for classical computing at increasing problem sizes. For moderate problem sizes, and a large number of samples, this full distribution calculation would likely be more performant than the current simulation. For larger problem sizes, chain-rule methods allow a better scaling in problem size at the cost of a worse scaling in sample number.

As mentioned previously, the simulation of a double-loop time-bin interferometer has run-time prohibitive for producing a full set of comparable data. From the structure of the algorithm, we know that it must be the sampling time that increases dramatically with the double-loop simulation. This is shown in Figure 7 over a range of small problems.

As seen in equation 3 and Figure 2, the cost function needs to be evaluated once for every sample within every iteration. If this is a costly (in time) calculation (if t_C is large compared to t_S), it will have a great impact on the total runtime. Therefore it is worth better understanding how this scales with n . Wall-clock runtime varies greatly depending on the background processes being run on the device and therefore, we have chosen to measure the clock cycles required for a single cost function evaluation for a

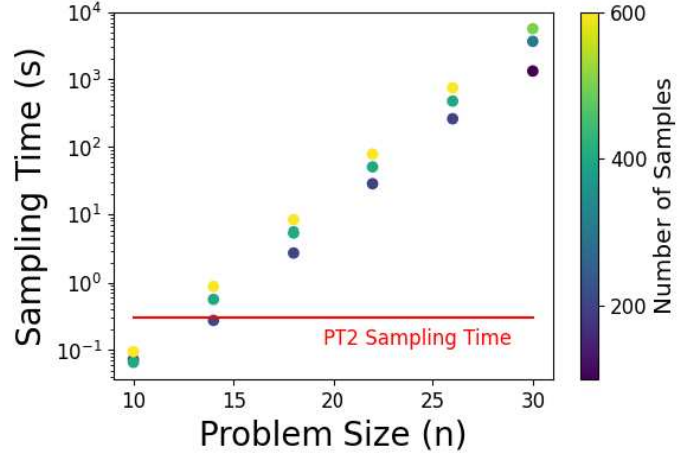


Fig. 7: Time taken to simulate a double loop time-bin interferometer for a range of problem sizes and draw range of samples. Note the log scale on the y-axis when comparing to Figure 6.

range of problem sizes. This is shown in Figure 8. Wall-clock time is assumed to follow the same scaling but with greater variability and magnitudes based on the capability of the CPU. From this result, it is clear to see that simplifying and optimising the cost function should be an area of focus in any practical use of the BBS algorithm, especially when considering large problems. Calculating the cost function for each sample within a single iteration of the BBS algorithm is an entirely parallelisable problem. Once all of the samples have been obtained from the device, each cost function evaluation can happen independently, which provides the potential for speed up depending on the quantity of compute resources allocated to the task.

Given the total number of evaluations of the cost function for a complete run of the BBS algorithm is $NSamp \times NIter$ (assuming all unique bit strings), it would be worth a deeper study into the balance between the chosen value of $NSamp$ and $NIter$. A large enough number of samples is required to “well map” the area of the search space that the current parameters provide, and to remain robust to noise and errors in the system. A large enough number of iterations allows for more parameter combinations to be tested, thus providing a broader exploration of the search space. Exploring this trade-off would be a valuable avenue of future work.

5 Conclusions and Future Work

At small problem sizes, the ORCA PT-2 device running the gradient-free Binary Bosonic Solver (BBS) finds dominating sets of equivalent size to classical methods. Within the range explored, at larger problem sizes, higher n , the performance of the PT-2 (and simulator) degrades compared to the classical methods. We hypothesise that the number of samples and/or iterations is currently restricting performance of the BBS at high n , and further work should look to exploring the effects of changing both

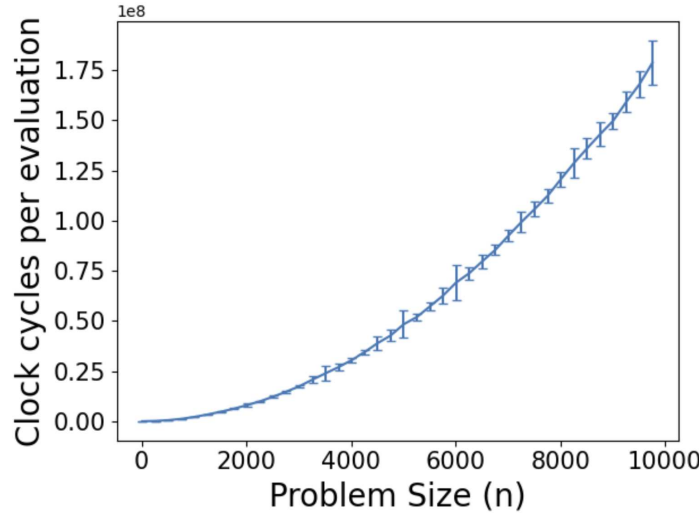


Fig. 8: The number of CPU clock cycles taken to calculate a single instance of the cost function against the problem size. The problem size represents the number of nodes in the graph. All graphs were produced with a connection density $p = 0.05$. For each instance, 10 unique calculations are performed. The plot shows the mean and standard deviation across these calculations.

the number of samples and number of iterations, and the possible trade-off between them.

With respect to the runtime of the whole algorithm, the largest factors in determining the runtime (using the PT-2 device) are the number of samples, the number of iterations, and the calculation of the cost function. As mentioned above, the number of samples and iterations should both be made as large as possible to allow the greatest possible exploration of the search space. Therefore, the most promising ways to optimise the runtime would be (a) to determine the minimum viable number of samples and iterations to achieve the required solution quality and/or allow the algorithm to terminate at a required solution quality, and (b) to optimise the design and parallelisation of the cost function calculations.

It would also be beneficial to consider the use of a hybrid algorithm of gradient-free initial “wide area” search and then gradient-based fine tuning to a better solution. This may allow a user to benefit from the computational efficiency of the gradient-free approach whilst maximising the likelihood of finding a minimal solution.

Declarations

Ethical Approval

Not applicable, no human or animal research involved.

Competing Interests

The authors have no competing interests of a financial or personal nature, nor other interests that might be perceived to influence the results and/or discussion reported in this paper.

Authors' contributions

J.P. wrote the first draft of the manuscript and performed all simulation experiments. S.S. and I.D'A. reviewed the manuscript, and contributed to the design of the experiments and analysis of results.

Funding

The authors wish to acknowledge Defence Science Technical Laboratory (Dstl) who are funding this research. Content includes material subject to © Crown copyright (2026), Dstl. This material is licensed under the terms of the Open Government Licence except where otherwise stated. To view this licence, visit <http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3> or write to the Information Policy Team, The National Archives, Kew, London TW9 4DU, or email: psi@nationalarchives.gov.uk

Data Availability

Supporting information is available from the authors.

Acknowledgments

Access to a PT-2 machine has been facilitated by ORCA Computing. ORCA Computing have also been very helpful in the discussion and understanding of the hardware details.

References

- Aaronson, S., & Arkhipov, A. (2011). The computational complexity of linear optics. *Proceedings of the 43rd annual acm symposium on theory of computing, stoc'11* (pp. 333–342).
- Arrazola, J.M., & Bromley, T.R. (2018). Using gaussian boson sampling to find dense subgraphs. *Phys. Rev. Lett.*, *121*(3), 030503,
- Bacarreza, O., Farnsworth, T., Makarovskiy, A., Wallner, H., Hicks, T., Sempere-Llagostera, S., ... Clements, W.R. (2025, August). Quantum latent distributions in deep generative models. *arXiv [cs.LG]*, , [cs.LG]
- Bradler, K., & Wallner, H. (2021). Certain properties and applications of shallow bosonic circuits. *arXiv:2112.09766 [quant-ph]*, ,

[quant-ph]

- Bromley, T.R., Arrazola, J.M., Jahangiri, S., Izaac, J., Quesada, N., Gran, A.D., ... Killoran, N. (2020). Applications of near-term photonic quantum computers: software and algorithms. *Quantum Sci. Technol.*, 5(3), 034010,
- Bulmer, J.F.F., Bell, B.A., Chadwick, R.S., Jones, A.E., Moise, D., Rigazzi, A., ... Laing, A. (2022, January). The boundary for quantum advantage in gaussian boson sampling. *Sci. Adv.*, 8(4), eabl9236,
- Deshpande, A., Mehta, A., Vincent, T., Quesada, N., Hinsche, M., Ioannou, M., ... Dhand, I. (2022, January). Quantum computational advantage via high-dimensional gaussian boson sampling. *Sci. Adv.*, 8(1), eabi7894,
- Grandoni, F. (2006). A note on the complexity of minimum dominating set. *J. Discrete Algorithms (Amst.)*, 4(2), 209–214,
- He, Y., Ding, X., Su, Z.-E., Huang, H.-L., Qin, J., Wang, C., ... Pan, J.-W. (2017). Time-bin-encoded boson sampling with a single-photon device. *Phys. Rev. Lett.*, 118(19), 190501,
- Hong, C.K., Ou, Z.Y., Mandel, L. (1987). Measurement of subpicosecond time intervals between two photons by interference. *Phys. Rev. Lett.*, 59(18), 2044–2046,
- Makarovskiy, A., Slys, M., Grodzki, L., Siera, D., Farnsworth, T., Clements, W.R., ... Kurowski, K. (2025, October). A binary optimisation algorithm for near-term photonic quantum processors. *arXiv [quant-ph]*, ,
[quant-ph]
- Motes, K.R., Gilchrist, A., Dowling, J.P., Rohde, P.P. (2014). Scalable boson sampling with time-bin encoding using a loop-based architecture. *Phys. Rev. Lett.*, 113(12), 120501,
- Novák, S., Roberts, D.D., Makarovskiy, A., García-Patrón, R., Clements, W.R. (2025, November). Boundaries for quantum advantage with single photons and loop-based time-bin interferometers. *Quantum*, 9(1915), 1915,
- Olson, J.P., Seshadreesan, K.P., Motes, K.R., Rohde, P.P., Dowling, J.P. (2015, February). Sampling arbitrary photon-added or photon-subtracted squeezed states is

- in the same complexity class as boson sampling. *Phys. Rev. A*, *91*(2), 022317,
- Park, J., Stepney, S., D’Amico, I. (2026). Benchmarking ORCA PT-1 boson sampler in simulation. *Lecture notes in computer science* (pp. 73–87). Cham.
- Sempere-Llagostera, S., Patel, R.B., Walmsley, I.A., Kolthammer, W.S. (2022, September). Experimentally finding dense subgraphs using a time-bin encoded gaussian boson sampling device. *Phys. Rev. X*, *12*(3), 031045,
- Slysz, M., Kurowski, K., Waligóra, G. (2024). Solving combinatorial optimization problems on a photonic quantum computer. *arXiv:2409.13781 [quant-ph]*, , [quant-ph]
- Teytaud, O., & Rapin, J. (2018, December). *Nevergrad: An open source tool for derivative-free optimization*. <https://engineering.fb.com/2018/12/20/ai-research/nevergrad/>. Meta. (Accessed: 2026-1-22)