Heuristic compactness maximization algorithm for two-dimensional single-atom traps rearrangement

T Mamee^{1,2,3}**, W Anukool**^{1,2,4} **, N Thaicharoen**^{1}**, N Chattrapiban**^{1,2,4} **and P Sompet**^{1,*}

^{1} Research Center for Quantum Technology, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand

^{2} Department of Physics and Materials Science, Faculty of Science, Chiang Mai University, Chiang Mai 50200, Thailand

^{3} Graduate School, Chiang Mai University, Chiang Mai 50200, Thailand

^{4} Thailand Center of Excellence in Physics, Commission on Higher Education, Bangkok 10400, Thailand

* E-mail: pimonpan.sompet@cmu.ac.th

Abstract. We establish an algorithm and computational results based on heuristic rearrangement of randomly filled array toward a defect-free and compact array. In this approach, the vacancies are filled from the inner layer that is related to the distance from the center of each loading site. By rearranging the position of atoms that maximize the compactness of the system layer by layer, the algorithm is set to iterate until the compactness reaches its local maximum. The results show that by applying the algorithm, the compactness of the system converges up to ∼97% of the theoretical maximum.

1. Introduction

The construction of defect-free arrays of single atoms is a basis towards synthesizing a fully controllable and scalable quantum system in terms of interaction and a number of interacting systems. The system is promising for applications such as quantum information processing [1], quantum simulation [2], quantum error correction [3] and quantum communication [4]. However, there is a limitation of constructing a perfectly filled deterministic array of single-atoms due to the collisional pair-loss induced by light-assisted collision [5] during a single-atom loading process. In this case, the loading efficiency is bounded to the success probability of ∼50% in the case of a large red-detuning of the assisted-collision light [5]. This probabilistic loading undoubtedly creates some vacancies in an initially loaded array. Thus, in order to obtain a defect-free array, ones can increase the loading efficiency via a more costly and complicated techniques e.g. those that employ blue-detuned laser light to induce light assisted collisions [6]. Another method is to rearrange filled traps to remove defected sites from the target area (pre-identified by fluorescence imaging) [7]. The rearrangement could be accomplished by transporting each atom one-by-one with a deep optical envelope via a 2d acousto-optic deflector (AOD) [7–9] or by changing the position of each individual trap inside an array with a spatial light modulator (SLM) via an optical-phase control [10]. The main focus of this work is in the later approach, the single-atom trap rearrangement.

Regarding to the capability of creating a fully-filled array, there are three key limitations constraining the rearrangement process, which are 1) a finite vacuum-limited lifetime of a trapped atom, 2) the heating mechanism during the trap translation, and 3) the trap size and separation. In particular, the vacuum-limited lifetime limits the upper bound of the total time an algorithm takes to calculate and rearrange traps. Concomitantly, the heating mechanism limits the number of all moves an atom can undergo before kinetic energy overcomes the trapping potentials. Lastly, the trap size and separation prohibit the translation along diagonals of the trapping array due to the number loss affected by a merged-trap geometry. Therefore, the atom-rearrangement algorithm is restricted to only translations along the edge of the array (a line connecting trap centers) and all the mentioned limitations must be taken into consideration.

In principle, an optimization algorithm [11] can be designed from a brute-force method, i.e. comparing total costs of all possible trajectories regardless of computation costs. However, due to the key limitations mentioned, the calculation of all possible trajectories is highly time consuming and therefore impractical [9]. Other alternatives that offer more practical calculation time and obtain nearly optimized solutions, concern a heuristic paradigm to solving problems of rearrangement from a specific set of instructions. Examples are the short-move first algorithm [7], the compression algorithm [9], the A* searching algorithm (ASA) and the heuristic cluster algorithm (HCA) [12].

In this work, we show a preliminary demonstration of a heuristic algorithm that we developed called the heuristic compactness maximization algorithm. The algorithm focuses on compressing a trap distribution to maximally packed with filled traps to minimize a designated spatial cost function and thus maximize the "compactness" of the system. As a results, the algorithm can be used to maximize compactness of the system in any desired region by minimizing the variable where the performance depends on the array's size.

2. Theory and background: Heuristic compactness maximization algorithm

2.1. Overall idea of the algorithm

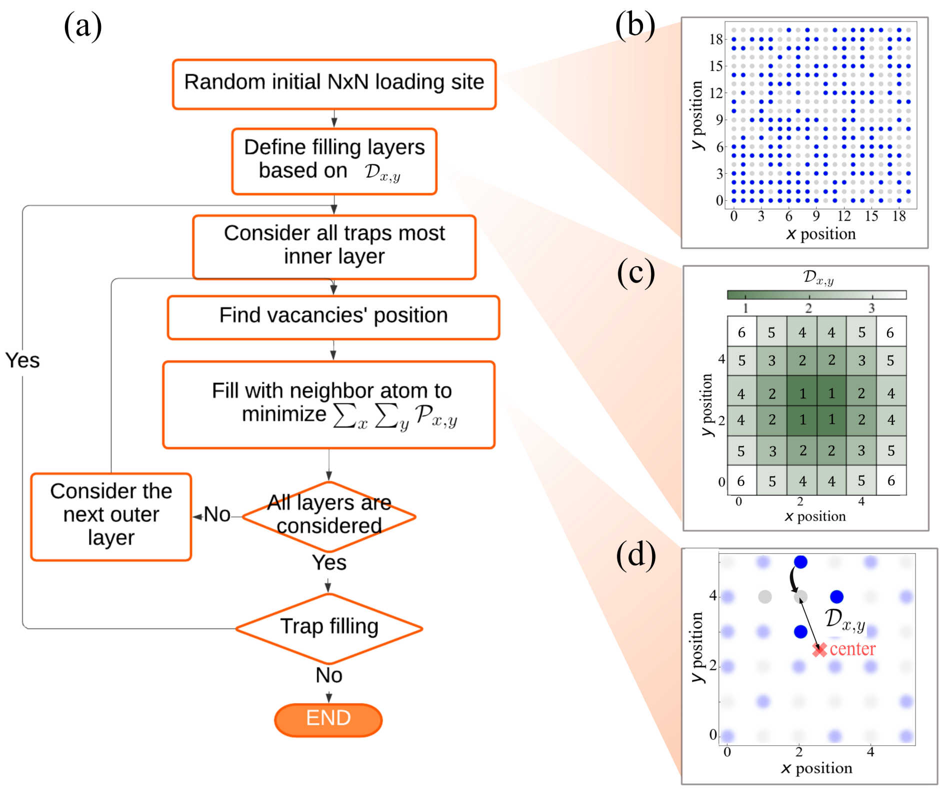
The main task of the heuristic compactness maximization algorithm is to maximize the compactness of the system defined through a moment matrix of filled-site $\mathcal{P} = \mathcal{O} \odot \mathcal{D}$, the element-wise multiplication between the occupancy matrix $\mathcal{O}$ and the distance matrix $\mathcal{D}$. The element of $\mathcal{P}$ at each array site $\{x, y\}$ is described by $\mathcal{P}_{x,y} = \mathcal{O}_{x,y}\mathcal{D}_{x,y}$, where $\mathcal{O}_{x,y}$ is the occupancy of a trap with value 1(0) for the occupied(unoccupied) trap, and the element $\mathcal{D}_{x,y}$ of $\mathcal{D}$ is defined as $\mathcal{D}_{x,y} = \sqrt{(x - x_0)^2 + (y - y_0)^2}$, where $\{x_0, y_0\}$ is the array center. The summation of all elements of $\mathcal{P}$ is

$$\mathcal{P}_{\mathrm{s}} = \sum_x \sum_y \mathcal{P}_{x,y}, \tag{1}$$

which represents the strength of the moment related to an average spreading of the occupied site from the array's center. Therefore, when the algorithm minimizes $\mathcal{P}_{\mathrm{s}}$, the compactness is maximized. As a result, the atoms seek the sites that have lower $\mathcal{P}_{x,y}$, which allows one to define the final shape of the filled system via the $\mathcal{P}$ matrix accordingly.

2.2. Algorithm flowchart

The flowchart of the algorithm is shown in figure 1(a). In the beginning, an initial traps loading is constructed from the Monte-Carlo method with 50-percent success probability to obtain a single atom for each trap. As a result, the temporal distribution of each single trap follows to a sub-Poissonian distribution. In this case, the filling factor is 0.5 [13] and this filling factor is used for all of the subsequent calculations throughout this work. To illustrate how the algorithm works, figure 1(b)–(d) are selected as examples. In figure 1(b) the algorithm generates a random distribution of an initial occupancy ($\mathcal{O}$ matrix). Afterward, it defines the filling layer shown in figure 1(c). In this example, there are 6 layers labeled in numbers. Each layer has the same value of $\mathcal{P}_{x,y}$. The algorithm starts its iteration loop from the smallest unfilled layer. Then it seeks to fill the vacancies in the layer with the nearest-neighbor atoms from the nearest outer layer. At this step, the algorithm aims to minimize the $\mathcal{P}_\mathrm{s}$ parameter. After the first layer is fully occupied, the next layer is considered. For an example as shown in figure 1(d), the vacancy at position $\{2,4\}$ needs to be filled. From here, the atom located in the next layer, i.e. at $\{2,5\}$, is selected because $\mathcal{P}_\mathrm{s}$ is the most minimized comparing to the other nearest neighbor atoms in the same layer. Finally, because the allowed trajectories for the atom is restricted to a move along the edges between traps (the step size is one between adjacent traps), the algorithm may need several steps until $\mathcal{P}_\mathrm{s}$ reaches the minimum.



(a)
Random initial NxN loading site
Define filling layers based on $\mathcal{D}_{x,y}$
Consider all traps most inner layer
Find vacancies' position
Fill with neighbor atom to minimize $\sum_x \sum_y \mathcal{P}_{x,y}$
All layers are considered
No
Consider the next outer layer
Yes
Trap filling
Yes
No
END
(b)
y position
x position
(c)
$\mathcal{D}_{x,y}$
y position
x position
(d)
y position
x position
$\mathcal{D}_{x,y}$
center

Figure 1. A schematic diagram of compactness maximization algorithm. From the flowchart (a), initial loaded array is randomized as shown in (b) and then the algorithm defines filling layers (shown in (c)) corresponding to each trap's specific spatial property called $\mathcal{D}_{x,y}$. To maximize the compactness, the partially filled innermost layer is assigned to the lowest $\mathcal{D}_{x,y}$. Once a vacancy is found, the only one nearest neighbor atom that can minimize $\mathcal{P}_s = \sum_x \sum_y \mathcal{P}_{x,y}$, as shown in (d), is moved. From here, the algorithm tends to fill the inner vacancies with atoms from outer layer where $\mathcal{D}_{x,y}$ is higher. (run over 500 loading samples)

3. Central occupancy evolution

In this section, the algorithm based on $\mathcal{P}_\mathrm{s}$ as defined in equation (1), is applied to the array of 20×20 loading sites. The calculation repeats over $k = 500$ randomly loaded $\mathcal{O}$ matrices to inquire about the averaged occupancy matrix, $\langle\mathcal{O}\rangle = \sum_{i=1}^{k} O_i/k$. From figure 2, at an initial setup (0^{th} iteration), the average occupancy $\langle\mathcal{O}^{(0)}\rangle$ of the system is nearly 0.5 and equally distributed over the loading sites. As the number of iterations increases to j, the average occupancy $\langle\mathcal{O}^{(j)}\rangle$ near the center of the array become higher and reach the saturated value of one, which is the maximum obtainable averaged occupancy because each site can only trap zero or one atom. After several iterations, the algorithm drive the system to form an x-shape pattern of saturated occupancy around the center. This is because the atoms are allowed to move only in either the x- or y- direction and the occupancy along the array's diagonals are populated from the moves in both directions. For a larger number of iterations, $\langle\mathcal{O}\rangle$ gradually increases toward uniform saturation, starting from the inner layer to the outer layer. The average occupancy is almost depleted near the boundary, forming a circular pattern respective to the defined $\mathcal{D}$ matrix. This compact round shape can be seen in the single-shot sample showing that all vacancies in the inner layer are eliminated while there is no atom left in the outer layer. This demonstrates the mass transportation driven by the algorithm from the wings toward the low $\mathcal{D}_{x,y}$ region around the center of the system.

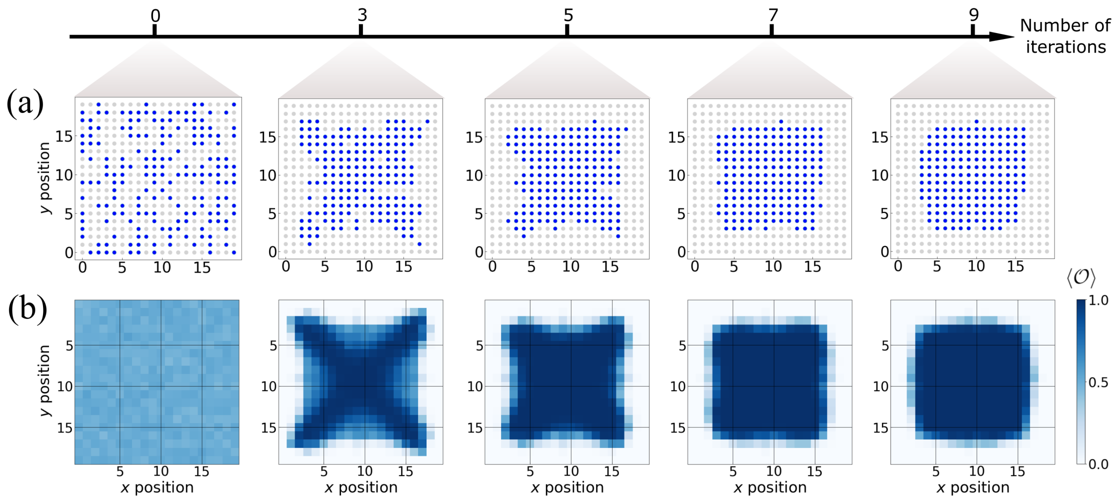


Figure 2. (a) Single-shot examples of $\mathcal{O}$ and (b) average occupancy matrices $\langle\mathcal{O}\rangle$ for each chosen number of iteration, showing compactness progression of a single-atoms array of 20×20 loading site, performed rearranging via $\mathcal{P}_{\mathrm{s}}$ based algorithm for radial $\mathcal{D}$.

4. Convergence of $\mathcal{P}_{\mathrm{s}}$ parameter

In this section, the behavior of the algorithm based on $\mathcal{P}_{\mathrm{s}}$ is established. The evolution of $\mathcal{P}_{\mathrm{s}}$ is shown in figure 3. Figure 3(a) shows that the average of $\mathcal{P}_{\mathrm{s}}$ exponentially decreases toward a saturated value as the number of iterations increases. This behavior agrees well with figure 2 where the system's compactness rises. For the 0^{th} iteration, the initial $\mathcal{P}_{\mathrm{s}}$ parameter is calculated from the condition that the atom is randomly distributed. For the later iterations, the value of $\mathcal{P}_{\mathrm{s}}$ gradually decreases until it converges to a local minimum and the system reaches local maximum of compactness. The ratio between the local maximum compactness $\mathcal{C}_{\mathrm{l}}$ and the theoretical compactness $\mathcal{C}_{\mathrm{th}}$ is calculated as follows:

$$\frac{\mathcal{C}_{\mathrm{l}}}{\mathcal{C}_{\mathrm{th}}} = \frac{\max(\mathcal{P}_{\mathrm{s}}) - \min(\mathcal{P}_{\mathrm{s}})}{\max(\mathcal{P}_{\mathrm{s}}) - \min(\mathcal{P}'_{\mathrm{s}})}, \tag{2}$$

where $\min(\mathcal{P}'_{\rm s})$ is the theoretical minimum of $\mathcal{P}_{\rm s}$. From the calculation of over 500 loading samples, $\langle \mathcal{C}_{\rm l}/\mathcal{C}_{\rm th} \rangle = 0.97$. The time constant τ of the decay is plotted in the insets inferring that the saturation time depends on the the system size. This could be further explained in figure 3(b) in which the inset shows the average of accumulated moves $\langle M_a \rangle$ for each iteration. The value of $\langle M_a \rangle$ moderately increases until it reaches an asymptotic value, indicating that the total number of the required moves for every atom in the system reaches the maximum compactness. Meanwhile, the plot between the maximum averaged value of the accumulated moves and the loading array's size in the main plot in figure 3(b) shows a nonlinear trend. For a larger loading array's size, more moves are needed to compact the system. This agrees well with a smaller decay rate for a larger array in figure 3(a).

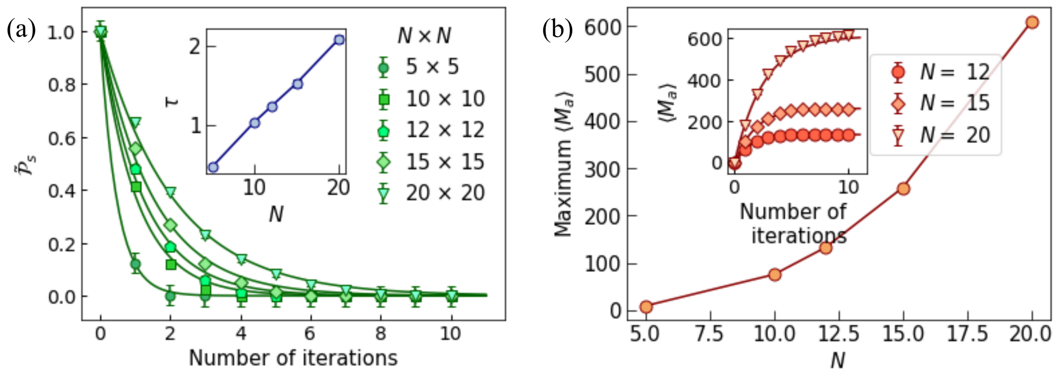


Figure 3. Evolution of $\mathcal{P}_{\rm s}$ and the effect of the system size. (a) The scaled $\mathcal{P}_{\rm s}$ (from 0 to 1), $\tilde{\mathcal{P}}_{\rm s}$, is plotted as a function of number of iterations for different array sizes. In the plot, $\tilde{\mathcal{P}}_{\rm s}$ is scaled with respect to the maximum value of $\mathcal{P}_{\rm s}$, $\max(\mathcal{P}_{\rm s})$, and the local minimum, $\min(\mathcal{P}_{\rm s})$, for each array size condition. Thus $\tilde{\mathcal{P}}_{\rm s}$ does not show the discrepancy between $\min(\mathcal{P}_{\rm s})$, and the expected theoretical minimum, $\min(\mathcal{P}'_{\rm s})$. The inset shows the decay constant τ from the exponential fit, plotted against the array lateral length N. (b) Maximum number of steps needed in the atom rearrangement algorithm for different lateral array lengths. The inset shows an accumulated number of moves, $M_{\rm a}$, increasing for each iteration. The data are averaged over 500 samples and each error bar represents one standard deviation, which is especially small in (b).

5. Arbitrary occupancy shaping

To demonstrate applicability of our algorithm, we provide three spatial distributions. In particular, the arrangement of trap potentially implies different physics that experimenters convey. In this section, to examine if the algorithm could give different trap configurations, three different $\mathcal{D}$ matrices are used in the algorithm. Figure 4 shows the averaged occupancy matrices $\langle \mathcal{O} \rangle$, the distribution matrices $\mathcal{D}$ and their line cross-section from left to right respectively. In the same figure, from top to bottom row, the $\mathcal{D}$ matrices are radially linear distribution, radially quadratic distribution and horizontal triangular distribution respectively. For the case of linear and quadratic distribution, the averaged occupation matrices $\langle \mathcal{O} \rangle$ are similar and have shapes between a circular and a rectangular. This is because the algorithm restriction on the direction of moves compresses $\mathcal{O}$ toward a rectangular shape at an early stage and stall at the local maximum compactness as mentioned in section 4. Nonetheless, for the triangular $\mathcal{D}$ matrix, the averaged occupancy drastically follow the triangular shape.

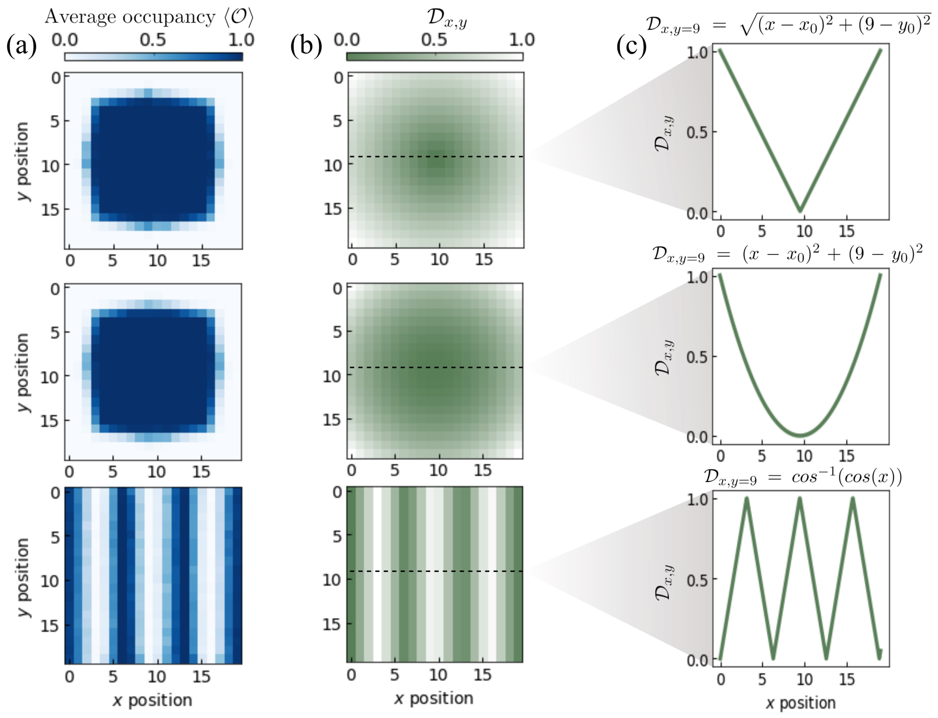


Figure 4. Averaged occupancy matrix $\langle\mathcal{O}\rangle$ (a) and $\mathcal{D}$ matrix (b) after performing the algorithm for three different $\mathcal{D}_{x,y}$ which alter the pattern of the layer structure. The cross section at $y = 9$ for each $\mathcal{D}$ matrix is shown in (c).

6. Conclusions

We propose the compactness maximization algorithm to rearrange atoms in a Cartesian array. The algorithm works by filling atom layer by layer from the innermost layer (based on $\mathcal{P}_\mathrm{s}$) to the outermost one while the averaged occupancy at the target region increases. The results shows that the system compactness could be maximized up to ∼97% of the theoretical maximum. For a larger array, more iteration is needed to reach the local maximum of compactness. Finally, the $\mathcal{D}$ matrix could be arbitrarily designed to give a desired atom distribution.

Several tasks are required to improve the performance of the algorithm. The main tasks relates to the optimization of moves and rearrangement time. In particular, coding efficiency and the local minimum escape can be improved by allowing complex trajectories via a reinforcement learning model.

References

[1] Saffman M, Walker T G and Mølmer K 2010 *Rev. Mod. Phys.* **82** 2313–63
[2] Feynman R P 1982 *Int. J. Theor. Phys.* **21** 467–88
[3] Fowler A G, Mariantoni M, Martinis J M and Cleland A N 2012 *Phys. Rev.* A **86**(3) 032324
[4] Duan L M, Lukin M D, Cirac J I and Zoller P 2001 *Nature* **414** 413–18
[5] Sompet P, Carpentier A V, Fung Y H, McGovern M and Andersen M F 2013 *Phys. Rev.* A **88**(5) 051401
[6] Carpentier A V, Fung Y H, Sompet P, Hilliard A J, Walker T G and Andersen M F 2013 *Laser Phys. Lett.* **10** 125501
[7] Barredo D, Léséleuc S D, Lienhard V, Lahaye T and Browaeys A 2016 *Science* **354** 1021–3
[8] Ebadi S *et al* 2021 *Nature* **595** 227–32

[9] Schymik K N, Lienhard V, Barredo D, Scholl P, Williams H, Browaeys A and Lahaye T 2020 *Phys. Rev.* A **102**(6) 063107

[10] Kim H, Lee W, Lee H g, Jo H, Song Y and Ahn J 2016 *Nat. Commun.* **7** 13317

[11] Lee W, Kim H and Ahn J 2017 *Phys. Rev.* A **95**(5) 053424

[12] Sheng C, Hou J, He X, Xu P, Wang K, Zhuang J, Li X, Liu M, Wang J and Zhan M 2021 *Phys. Rev. Res.* **3**(2) 023008

[13] Ohl de Mello D, Schäffner D, Werkmann J, Preuschoff T, Kohfahl L, Schlosser M and Birkl G 2019 *Phys. Rev. Lett.* **122**(20) 203601