NUCL SCI TECH (2019) 30:56
https://doi.org/10.1007/s41365-019-0582-6

GPU-accelerated scanning path optimization in particle cancer therapy

Chao Wu[1] · Yue-Hu Pu[1,2] · Xiao Zhang[2]

Received: 25 December 2017 / Revised: 19 July 2018 / Accepted: 20 October 2018 / Published online: 13 March 2019
© China Science Publishing & Media Ltd. (Science Press), Shanghai Institute of Applied Physics, the Chinese Academy of Sciences, Chinese Nuclear Society and Springer Nature Singapore Pte Ltd. 2019

Abstract When using the beam scanning method for particle beam therapy, the target volume is divided into many iso-energy slices and is irradiated slice by slice. Each slice may comprise thousands of discrete scanning beam positions. An optimized scanning path can decrease the transit dose and may bypass important organs. The minimization of the scanning path length can be considered as a variation of the traveling salesman problem; the simulated annealing algorithm is adopted to solve this problem. The initial scanning path is assumed as a simple zigzag path; subsequently, random searches for accepted new paths are performed through cost evaluation and criteria-based judging. To reduce the optimization time of a given slice, random searches are parallelized by employing thousands of threads. The simultaneous optimization of multiple slices is realized by using many thread blocks of general-purpose computing on graphics processing units hardware. Running on a computer with an Intel i7-4790 CPU and NVIDIA K2200 GPU, our new method required only 1.3 s to obtain optimized scanning paths with a total of 40 slices in typically studied cases. The procedure and optimization results of this new method are presented in this work.

Keywords Particle beam therapy · Treatment planning · Scanning path optimization

✉ Yue-Hu Pu
puyuehu@sinap.ac.cn

[1] Shanghai Institute of Applied Physics, Chinese Academy of Sciences, Shanghai 201800, China

[2] Shanghai APACTRON Particle Equipment Co. Ltd, Shanghai 201800, China

1 Introduction

In the past decade, there has been growing interest in cancer therapy using particle beams, especially proton and carbon beams [1]. There are 73 particle therapy facilities in operation and 45 new facilities under construction according to the latest data from the PTCOG. Scanning irradiation methods are utilized by most new particle therapy facilities. Compared with passive scattering and wobbling methods, scanning irradiation methods can avoid the cumbersome use of patient-specific compensators and collimators [2]. In the active beam scanning method, the narrow pencil beams are steered by a pair of deflection magnets in the lateral direction, while scanning irradiation of the target in the depth direction is usually achieved by varying the particle beam energy, either through range shifters installed in the irradiation nozzle, beam transportation lines, or by using the direct energy variation of a synchrotron [3, 4].

In scanning treatment plans, a target volume is divided into many iso-energy slices. Each of these slices is further divided into many discrete beam positions (beam spots), where an optimized number of particles are assigned by the treatment planning software [5]. The scanning method includes two categories: spot and raster scanning. During the irradiation of a given slice using the raster scanning method, the beam is not switched off while moving the beam from one beam spot or beam position to another beam spot, unlike spot scanning [6, 7]. In the raster scanning method, the beam is turned off when the irradiation of a given slice is finished and then the next beam energy value of the next slice to be irradiated is set by the control system. In general, the irradiation can be performed more efficiently using the raster scanning method compared with

the spot scanning method, as there is no frequent beam turning-on and turning-off operations during the irradiation process. The broader interest in raster scanning is mainly due to the faster repainting capability that it offers, which is important in the presence of moving targets or other cases where intra-fraction geometric variation exists [8–10]. In a raster scanning system, the dose delivered during the transition time is usually added to the dose delivered on the next beam spot through a smart dose control algorithm. Because the beam is not turned off during the irradiation of a given slice, except for an interlock situation, the transit particles delivered between planned beam positions along the scanning path are roughly proportional to the total path length [5]. Therefore, an optimized scanning path can significantly reduce the possible dose deviation. Moreover, each slice may comprise thousands of discrete beam positions. When the scanning spots pattern is relatively regular and simple, the demand for an optimized scanning path is relatively small and an ordinary zigzag path (back-and-forth and top-to-bottom succession) is sufficient. However, due to the inhomogeneity of tissues in front of the target and the influence caused by the shape of the target, in many clinical cases the scanning position patterns can be highly irregular and may contain many vacant areas where important organs may exist. An optimized scanning path can efficiently reduce the effect caused by transit particles to realize a better dose distribution and provide better protection of important organs [11]. In comparison with scattering and wobbling methods, treatment planning plays a more important role in patient treatment using beam scanning methods. Besides dose calculation and dose optimization, scanning path optimization in treatment planning can also contribute to maximizing the benefits of scanning methods.

The importance of scanning path optimization was discussed in two published studies [2, 11]. In both studies, the simulated annealing (SA) algorithm was chosen to find the optimized scanning paths. These optimized scanning paths were then compared with the original zigzag paths. In the study of Kang et al. [2], scanning path optimization based on the SA algorithm was tested on archived patient and study cases, including one prostate case and three head and neck cases. The results showed that the scanning path optimization method based on the SA algorithm yielded path lengths that were $\sim$ 13–56% shorter than those of the zigzag patterns [2]. Pardo et al. compared the delivered dose distributions obtained by zigzag scanning and optimized paths. The results showed an obvious reduction in transit dose when the optimized path was used. The reduction in transit dose can potentially allow for the use of higher beam intensities, thus decreasing the treatment time [11]. In all the previous studies, the scanning paths were optimized slice by slice. In clinical cases, the number of slices of the target volume may be up to one hundred, which means that optimizing scanning paths slice by slice is time-consuming. In the work of Kang et al. [2], which used MATLAB on a computer with a 3.2 GHz processor, the SA optimization program required approximately 6 min for one beam comprising 45 energy slices and a total number of 1650 spots. In the work of Pardo et al. [11], which used C++ on a computer with a 3.2 GHz processor, the SA optimization program required approximately 6 min for one beam comprising 38 energy slices and a total number of 6892 spots. In either case, it took more than 8 s on average for each slice.

The primary goal of this research is to provide a new method and computer implementation for scanning path optimization by taking advantage of the general-purpose computing on graphics processing units (GPGPU)-based parallel computing technique. The method is described in Sect. 2, and the results are presented in Sect. 3.

2 Experimental section

2.1 Scanning path optimization as a modified traveling salesman problem

The scanning path optimization problem is analogous to the famous traveling salesman problem (TSP) [12] and can be solved approximately using heuristic methods, such as the SA algorithm. Optimizing the scanning path of one iso-energy slice can be described as follows: Given a certain number of scanning positions and the distances between each pair of beam positions, find the least-cost scanning path passing through all of the planned scanning positions without repeated visits. In a raster scanning system, beam scanning is performed relatively more frequently in one direction (X) and less frequently in another (Y). In order to reduce the position errors caused by magnetic hysteresis, scanning is designed to begin from the top row (x direction) and ends at the bottom row of the scanning pattern of a slice. Furthermore, dissimilar to a typical TSP, scanning is not required to return to the starting position. Therefore, the cost to move from the scanning position A (x_1, y_1) to B (x_2, y_2) is then the Euclidean distance, where (x_1, y_1) and (x_2, y_2) are the Cartesian coordinates of grid points A and B in a given slice. Given the total number of scanning positions K, the scanning order is (X_1, Y_1), (X_2, Y_2) … (X_K, Y_K). Scanning position (X_1, Y_1) is the starting position and (X_K, Y_K) is the ending position. The total path length, f, of the scanning path, P, is given by [2]

$$f(P) = \sum_{n=1}^{K-1} \sqrt{(x_{n+1} - x_n)^2 + Q \times (y_{n+1} - y_n)^2}. \quad (1)$$

This equation is the objective function to be minimized. Q is a penalty factor for the scanning motion in the Y direction and is usually set to 1.0, as for this work. The Q value can be set higher than 1.0 to intentionally reduce scanning motion in the Y direction.

2.2 Scanning path optimization program based on the SA algorithm

The SA algorithm imitates the cooling and annealing processes of molten metal to find a global minimum of the objective function [13, 14]. It basically includes iterative random searching procedures characterized with predetermined accepting rules and scheduled cooling of the temperature parameter [15]. Similar to the work of Pardo et al. [11], a C++ program was developed. The main work flow of the program includes the generation of an initial zigzag path and the optimization of the scanning path through the cooling and annealing mimicking procedure [5]. The program starts with an initial temperature parameter (T_0 = 0.5), and then T0 is cooled down over 1000 steps toward zero. With each temperature parameter, the program executes 100 × N_s random searches. N_s is the total spot number of a given slice. At each search step, the cost value, f, is calculated and path updating is determined according to the probabilistic function characterized by the temperature parameter. This setting of operational parameters was effective for finding a satisfactory path solution.

There are several methods to generate a new scanning path from the current one. For simplicity, we only consider the method [16] shown schematically in Fig. 1; two scanning spots are randomly selected, and then the scanning orders of the spots between these two selected spots are reversed.

Figure 1a shows the initial zigzag scanning path. Assuming that spots 5 and 10 are randomly selected, the scanning orders of the spots between spots 5 and 10 need to be reversed. In the next step, the program connects spot 5 with spot 9, and spot 6 with spot 10, as shown in Fig. 1b. The program then cuts the connections between spots 5 and 6, and spots 9 and 10. The newly generated path is shown in Fig. 1c. The path difference between the paths in Fig. 1a, c can be computed as

$$\Delta l = L(5,9) + L(6,10) - L(5,6) - L(9,10). \quad (2)$$

Rearranging the scanning path during the random searches in this way provides the benefit of obtaining the path length change without excessive calculations. Furthermore, it allows for both large and small modifications of the path length. Large modifications are effective for accelerating the convergence in the initial stage and to escape from local minima, while small modifications are efficient in the proximity of the minimum [11].

2.3 Fast path optimization of a single slice with parallel random searches

After applying our developed path optimization program to a number of cases including those mentioned in the literature, we found that it often requires thousands of serial random searches before updating to an accepted new scanning path. We studied four clinical and two study cases mentioned in Kang's work [2] and analyzed the occurrence frequency of the required number of random searches, Ni, before updating to an accepted path. The result of this study is summarized in Table 1.

It can be observed in Table 1 that less than 10% of the accepted paths were obtained within 100 random searches and more than 50% of accepted paths were obtained over 1000 random searches in all six cases. The average numbers of random searches that the scanning path optimization program required before updating to an accepted new path are greater than 1756 in all six cases. More scanning paths of other patterns were tested, and the results were similar. These results demonstrate that the previous method of scanning path optimization based on the SA algorithm, as in the literature, requires large numbers of serial random

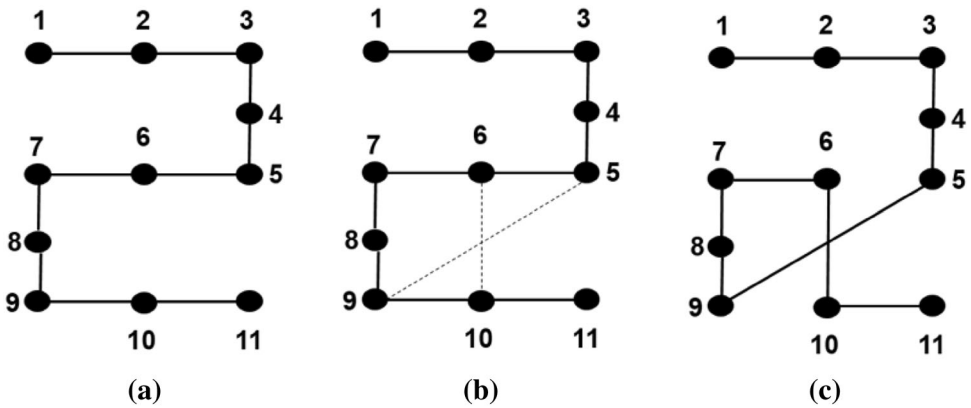

Fig. 1 Elaborate procedure to randomly generate a new scanning path. Spots 5 and 10 are randomly selected, and the scanning orders of the spots between these two selected spots are reversed

Table 1 Occurrence frequency of the required number of random searches, N_i, and the related statistics in all six studied cases

Case ID	Number of spots	$N_i \leq 100$ (%)	$N_i > 1000$ (%)	Average of N_i	Maximum of N_i
1	112	8.9	50.6	1757	20,089
2	145	6.6	57.1	2046	21,299
3	244	4.8	72.5	5472	63,003
4	218	5.5	68.0	3481	48,724
5	100	9.3	56.8	2709	34,823
6	108	8.4	64.4	3430	43,044

searches, which account for a major part of the optimization time. Therefore, it is a natural strategy to seek the possibility of parallelizing this random search process to substantially reduce the optimization time.

A brief description of our newly developed parallelized method of scanning path optimization based on the SA algorithm is shown in Fig. 2. In our new scheme, the serial random searches of previous works [2, 5, 11] are replaced with parallel random searches. This new process can be deemed as replacing the throwing of a single dice 1024 times with throwing 1024 dices at one time. CUDA parallel programming [17] is employed to realize these parallel random searches. The flowchart of a CUDA program using the parallel random search method is shown in Fig. 3.

In a CUDA parallel programming framework, computation tasks are categorized into CPU host and GPU device tasks which are executed by calling CUDA kernels. When the kernels are called on the host (CPU), the kernels start a thread grid and are executed N times in parallel by N different threads on the device (GPU) [17]. A thread grid consists of many thread blocks that are assigned for different tasks. A thread block comprises multiple threads. As shown in Fig. 3, in the new program, N_t threads simultaneously search for an accepted new path based on the same current path P0 in each iteration, and the maximum number of N_t is 1024 due to the limit of the GPU hardware. It should be noted that when N_t is set to 1, the parallel random search method is identical to the previous serial implementation found in the literature. When a parallel random search step is finished, one of the following three situations may occur: (1) No new accepted path is found and no path updating takes place; (2) only one new accepted path is found and the program updates the path; and (3) more than one new accepted path is found, and the program randomly chooses one accepted path to update. Then, the program repeats the search process following the previously described cooling and annealing procedure. After a certain number of parallel random search iterations, the ending conditions are satisfied and the program will then return to the current path and exit. In this developed program, random numbers are generated by the cuRAND library [17] to guarantee the equivalence of serial random and simultaneous random searches. Furthermore, this equivalence is strengthened by randomly selecting one accepted path when multiple accepted paths are found in one parallel random search iteration step.

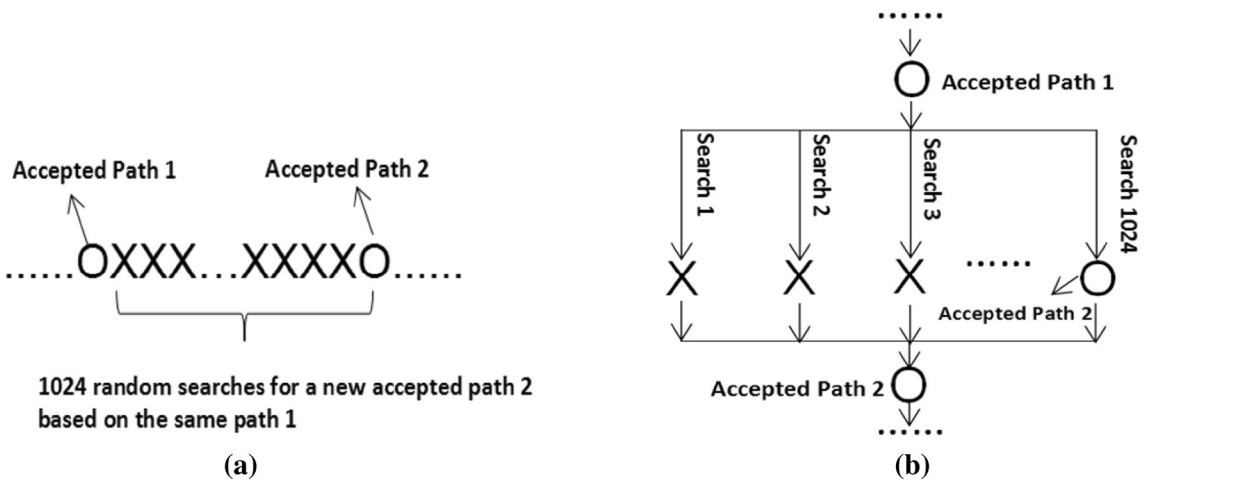

Fig. 2 Explanation of the method to reduce the optimization time of the scanning path optimization based on the SA algorithm by replacing large numbers of serial random searches with parallel random searches. **a** Examples of serial random searches. **b** Examples of parallel random searches

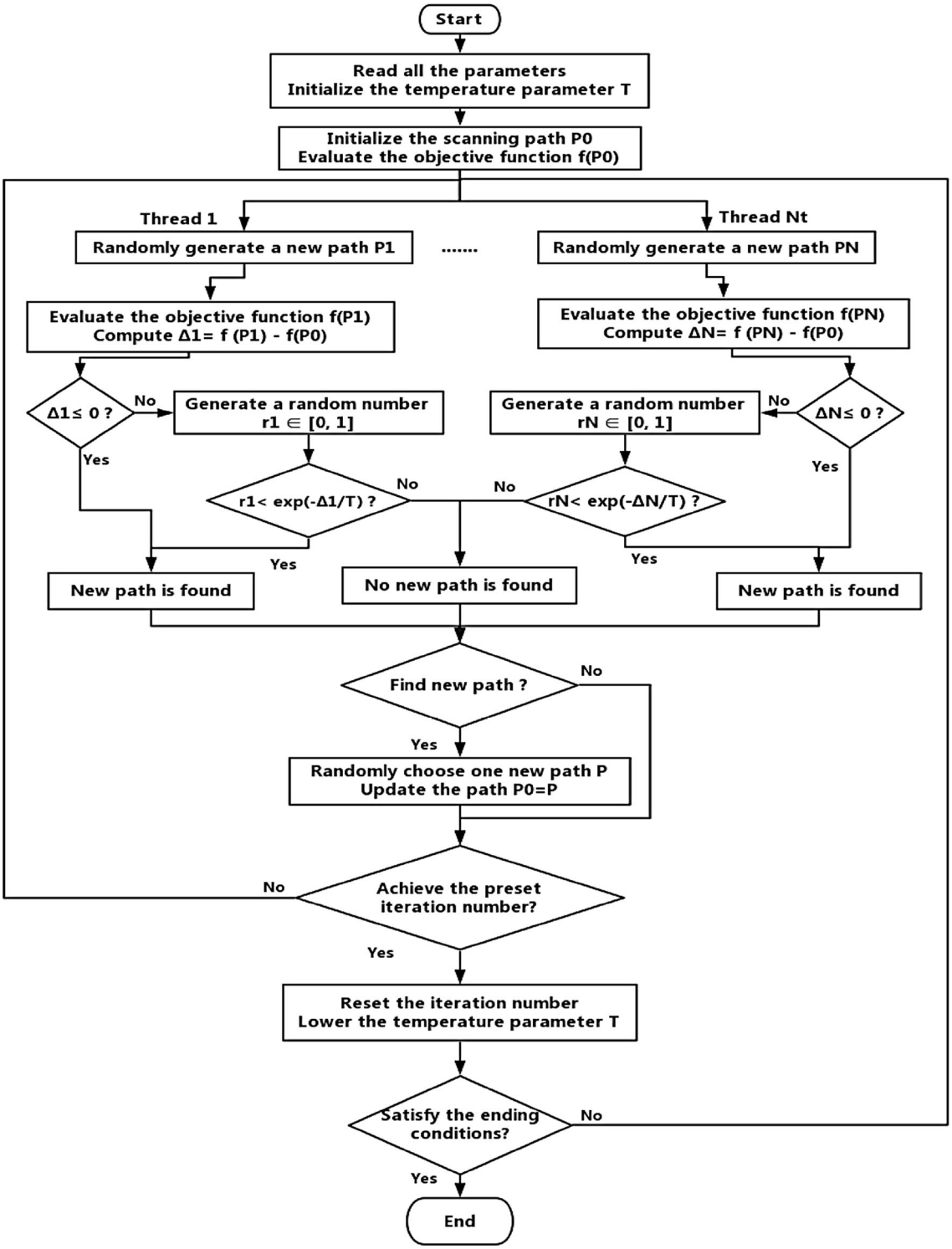

Fig. 3 Flowchart of a new program using parallel random search method taking advantage of CUDA programming

2.4 Simultaneous optimization of scanning paths of multiple slices

In the previous section, the optimization of a single slice using the parallel random search method is realized by employing thousands of threads of one thread block. Considering that the optimizing processes of the scanning paths in different slices are independent and have no demand for data transmission, the N_b thread blocks are employed for the simultaneous optimization of the

scanning paths of N_b slices. Therefore, each thread block handles one slice. The detailed optimization process in each thread block is identical to the process described in the previous section. The optimization across all blocks is concurrent but independent, so the scanning paths of multiple slices can be optimized in parallel. In clinical cases, the number of slices can be up to one hundred. For example, when the target thickness is 200 mm and the slice pitch is 2 mm, the number of slices is 100. Therefore, the simultaneous optimization of scanning paths of multiple slices can substantially reduce the optimization time.

3 Results

3.1 Results of path length reduction

The developed path optimization program was validated by applying it to clinical situation problems. We used the six cases mentioned in Kang's work as described in the above sections. The path length optimization results of all six different patterns are given in Table 2. The initial zigzag and optimized paths of two clinical cases are shown schematically in Figs. 4 and 5.

From Table 2 and Figs. 4 and 5, it is easily observed that the scanning path lengths are largely reduced and useless movements between the scanning positions are suppressed as expected. When the spatial arrangement of scanning spots is irregular and contains disperse clusters in the slice, the path length reduction can be greater than 50%. Besides these cases, various other complex patterns were also tested. All these show that the developed program using the parallel random search scheme is effective in reducing the scanning path lengths. Additionally, the path length reduction rates obtained in this work are found to be similar or even better than those disclosed in published works.

Table 2 Path length optimization results of the six studied cases. S_i, S_o, and ΔS denote the initial path length, optimized path length, and the reduction rate, respectively

Case ID	Case type	S_i (mm)	S_o (mm)	ΔS (%)
1	Head and neck	530	371	30.0
2	Head and neck	603	467	22.6
3	Study	1192	779	34.6
4	Study	1046	708	32.3
5	Prostate	1137	507	55.4
6	Prostate	1384	518	62.6

3.2 Results of the optimization time reduction

In order to verify the optimization time reduction produced by the parallelization of the random searches, we compared the calculation time using the single-thread CPU-based serial random and GPU-based parallel random search programs. The CPU used in the testing work is a 3.6 GHz Intel Core i7-4790 CPU. The CUDA program runs on a PC hosting a NVIDIA Quadro K2200 GPU with the same CPU. For each tested case, the program is executed ten times and the average optimization time is computed. The testing results of the six cases mentioned in Sect. 3.1 are shown in Table 3. It is observed that the parallel random search-based program is at most nine times faster than the serial random search-based program. The relation between the improved optimization time and number of employed threads for parallel random search is discussed in the next section.

The developed parallel random search program is then applied to simultaneously optimize the scanning paths of 40 slices. When the total spot number of all 40 slices is 4000, the optimization time is only 1.3 s using the same hardware mentioned above. Completing the same task with the same CPU hardware, the optimization time of the C++ single-thread program based on the conventional serial random search method is 61.3 s. These optimization results further demonstrate the advantage of the proposed GPU-based new parallel random search method.

4 Discussion

During the optimization process of the parallel random search method, because of the requirement of thread synchronization and memory access efficiency as well as the time needed for updating the paths after an accepted path is found, using N threads for parallel random searches does not cause an optimization speed increase by a factor of N. Given these limitations, the fact that our parallel search method still outperformed the conventional serial random search method by up to nine times means that our new method has obvious advantages in the path optimization of a single slice. In addition, the advantage of using GPU implementation for the parallel search method when optimization of multiple slices is needed is inarguable. Therefore, in this work CUDA programming using a GPU is adopted to realize the developed parallel random search method, while the method can also be realized using large numbers of CPUs.

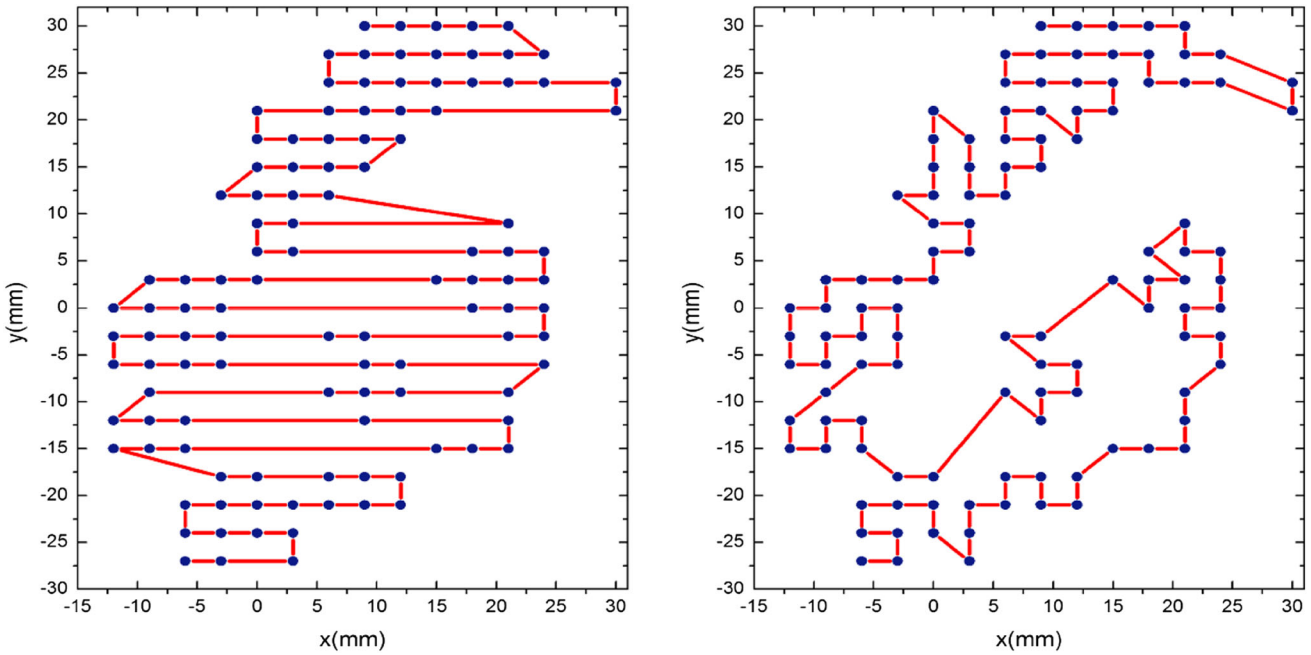

Fig. 4 Path length optimization result of a single slice found in case 1, a head and neck cancer patient. The left and right plots show the initial zigzag path and optimized path obtained in this work, respectively. The path length reduction is 30%

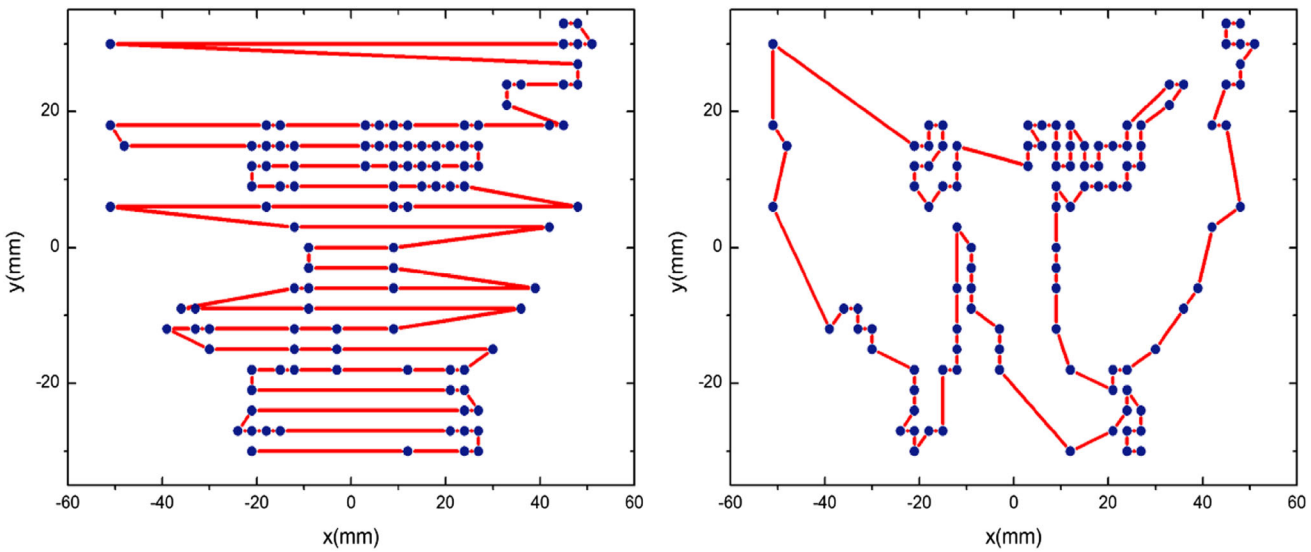

Fig. 5 Path length optimization result of a single slice found in case 6, a prostate cancer patient. The left plot shows the initial zigzag path; the right plot shows the optimized path obtained in this work. The path length reduction is 62.6%

Table 3 Average optimization time comparison of the conventional serial random search (T_s) and new parallel random search (T_p) methods

Case ID	T_s (s)	T_p (s)	T_s/T_p
1	1.639	0.217	7.6
2	2.121	0.249	8.5
3	3.502	0.404	8.7
4	3.171	0.395	8.0
5	1.431	0.173	8.3
6	1.592	0.176	9.0

5 Conclusion

A new and efficient parallel random search method for scanning path optimization was developed. The large numbers of serial random searches in the conventional scanning path optimization method based on the SA algorithm were replaced with parallel random searches. By applying the developed code to clinical data found in the literature, it was found that better scanning paths and

shorter optimization times can be achieved. The CUDA program developed in this work is capable of simultaneously optimizing as many as 100 slices of the scanning paths, which meets the clinical requirement of particle beam therapy, and the optimization speed is fast enough for achieving real-time optimization. The developed path optimization code can be integrated with a particle beam treatment planning system.

References

1. Y.H. Yang, M.Z. Zhang, D.M. Li, Simulation study of slow extraction for the Shanghai Advanced Proton Therapy facility. Nucl. Sci. Tech. **28**, 120 (2017). https://doi.org/10.1007/s41365-017-0273-0
2. J.H. Kang, J.J. Wilkens et al., Demonstration of scan path optimization in proton therapy. Med. Phys. **34**(9), 3457–3464 (2007). https://doi.org/10.1118/1.2760025
3. T. Inaniwa, T. Furukawa et al., Evaluation of hybrid depth scanning for carbon-ion radiotherapy. Med. Phys. **39**(5), 2820–2825 (2012). https://doi.org/10.1118/1.4705357
4. Th Haberer, W. Becher, D. Schardt et al., Magnetic scanning system for heavy ion therapy. Nucl. Instrum Methods **330**(1–2), 296–305 (1993). https://doi.org/10.1016/0168-9002(93)91335-K
5. M.F. Dias, M. Riboldi, J. Seco et al., Scan path optimization with/without clustering for active beam delivery in charged particle therapy. Physica Med. **31**, 130–136 (2015). https://doi.org/10.1016/j.ejmp.2015.01.001
6. T. Inaniwa, T. Furukawa, T. Tomitani et al., Optimization for fast-scanning irradiation in particle therapy. Med. Phys. **34**, 3302–3311 (2007). https://doi.org/10.1118/1.2754058
7. T. Furukawa, T. Inaniwa, S. Sato, T. Shirai et al., Performance of the NIRS fast scanning system for heavy-ion radiotherapy. Med. Phys. **37**, 5672–5682 (2010). https://doi.org/10.1118/1.3501313
8. S.M. Zenklusen, E. Pedroni, D. Meer, A study on repainting strategies for treating moderately moving targets with proton pencil beam scanning at the new gantry 2 at psi. Phys. Med. Biol. **55**, 5103–5121 (2010). https://doi.org/10.1118/1.3501313
9. C. Bert, M. Durante, Particle radiosurgery: a new frontier of physics in medicine. Phys. Medica **30**(5), 535–538 (2014). https://doi.org/10.1016/j.ejmp.2014.04.011
10. A.C. Knopf, A.J. Lomax, In the context of radiosurgery-Pros and cons of rescanning as a solution for treating moving targets with scanned particle beams. Phys. Medica **30**(5), 551–554 (2014). https://doi.org/10.1016/j.ejmp.2014.03.010
11. J. Pardo, M. Donetti, F. Bourhaleb et al., Heuristic optimization of the scanning path of particle therapy Beams. Med. Phys. **36**(6), 2043–2051 (2009). https://doi.org/10.1118/1.3121506
12. C. Voudouris, E. Tsang, Guided local search and its application to the traveling salesman problem. Eur. J. Oper. Res. **113**(2), 469–499 (1999). https://doi.org/10.1016/S0377-2217(98)00099-X
13. S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing. Science **220**, 671–680 (1983). https://doi.org/10.1126/science.220.4598.671
14. Y. Xiang, S. Gubian, G. Suomela et al., Generalized simulated annealing for global optimization: the GenSA package. J. Mech. Eng. **5**(1), 153–160 (2013). https://doi.org/10.32614/RJ-2013-002
15. E.R. Correia, V.B. Nascimento, C.M. de Castilho et al., The generalized simulated annealing algorithm in the low energy electron diffraction search problem. J. Phys.: Condens. Matter **17**(1), 1–16 (2005). https://doi.org/10.1088/0953-8984/17/1/001
16. V.T. Vetterling, W.H. Press, S.A. Teukolsky, B.P. Flannery, *Numerical Recipes Example Book (C++): The Art of Scientific Computing* (Cambridge University Press, Cambridge, 2002)
17. NVIDIA Corporation. CUDA C programming guide. http://docs.nvidia.com/cuda/cuda-c-programming-guide/#axzz4lLlZaXLb. Accessed 10 Apr 2017