Automation in Construction 62 (2016) 133–147

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon

Review

# Parallel genetic algorithm based automatic path planning for crane lifting in complex environments

Panpan Cai [a], Yiyu Cai [a,c,*], Indhumathi Chandrasekaran [a], Jianmin Zheng [b,c]

[a] School of Mechanical and Aerospace Engineering, Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore
[b] School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, 639798, Singapore
[c] Institute for Media Innovation, Nanyang Technological University, 50 Nanyang Drive, Research Techno Plaza, 637553, Singapore

ARTICLE INFO

*Article history:*
Received 18 December 2014
Received in revised form 8 September 2015
Accepted 20 September 2015
Available online 10 December 2015

*Keywords:*
Crane lifting path planning
Parallel genetic algorithm
Image-space collision detection
GPU computing
Hybrid C-space method

ABSTRACT

Heavy lifting is a common and important task in industrial plants. It is conducted frequently during the time of plant construction, maintenance shutdown and new equipment installation. To find a safe and cost effective way of lifting, a team works for weeks or even months doing site investigation, planning and evaluations. This paper considers the lifting path planning problem for terrain cranes in complex environments. The lifting path planning problem takes inputs such as the plant environment, crane mechanical data, crane position, start and end lifting configurations to generate the optimal lifting path by evaluating costs and safety risks. We formulate the crane lifting path planning as a multi-objective nonlinear integer optimization problem with implicit constraints. It aims to optimize the energy cost, time cost and human operation conformity of the lifting path under constraints of collision avoidance and operational limitations. To solve the optimization problem, we design a Master–Slave Parallel Genetic Algorithm and implement the algorithm on Graphics Processing Units using CUDA programming. In order to handle complex plants, we propose a collision detection strategy using hybrid configuration spaces based on an image-based collision detection algorithm. The results show that the method can efficiently generate high quality lifting paths in complex environments.



**Contents**



* Corresponding author.
*E-mail addresses:* pcai2@e.ntu.edu.sg (P. Cai), MYYCAI@ntu.edu.sg (Y. Cai), Chandra@ntu.edu.sg (I. Chandrasekaran), ASJMZheng@ntu.edu.sg (J. Zheng).

http://dx.doi.org/10.1016/j.autcon.2015.09.007
0926-5805/

## 1. Introduction

Heavy lift planning is an important job in industrial plants. To lift large and heavy targets, capacities of cranes can reach up to thousand tons. Lifting operations, however, work with potential accidents. The OSHA database [1] till year 2013 reported a total 3135 crane related accidents in USA. Many of them caused human death. Lifting safety is thus of utmost importance. Cost reduction is another major concern in the lifting industry. According to a rental rate survey conducted by Cranes & Access in 2011 [2], the average daily rental of a terrain crane with a capacity of 350 tons can cost almost US$ 8400. Electric power and fuel consumptions also take up a significant portion of the cost in lifting operations. For instance, the power consumption of the terrain crane LTM 1200 from Liebherr is 370 kW per hour [3]. Therefore, optimizing the time and energy cost in crane usage is highly desired. Besides, manpower cost is increasingly becoming a critical factor in lifting. The lifting path planning involves a complicated and sophisticated decision making process conducted by a lifting team. The team consists typically of one lifting supervisor or manager, one engineer, one crane operator, one or more signalmen and riggers. Currently, lifting path planning is mostly done by manual exercise which can be error-prone and very time-consuming. Even with an experienced lifting team, it easily takes a few weeks to complete the entire planning procedure.

Heavy lift planning involves multiple phases typically including crane selection, crane location determination, and lifting path planning. For lifting projects with multiple lifts, the planning may involve scheduling taking into consideration the interference among multiple cranes. If the plant contains dynamic objects, the planning also requires replanning mechanisms to alter the lifting path according to the changed environment. There are different types of commonly used cranes: tower cranes, terrain cranes, crawler cranes and so on. Among them, tower cranes have the least number of Degrees of Freedom (DOFs) and crawler cranes have most DOFs (up to 7). This paper focuses on the lifting path planning problem for terrain cranes with the assumption that scheduling, crane selection, crane locating and feasibility checking are readily available. Our objective is to develop an automatic lifting path planning system being able to output optimized lifting paths in near real-time and thus improve the safety and efficiency of heavy lifting in complex environments such as petrochemical and pharmaceutical plants, and construction sites.

### 1.1. Computer-aided lift planning

The complexity and hazards of lifting operations inspire the development of computer-aided lift planning methods making use of computer simulations and computations to assist the lift planning process. Early efforts focused on the use of simulation based systems to assist interactive lift planning. These systems helped in automated mechanical checking, safety monitoring and evaluations for interactive lifting paths. Hornaday et al. [4] proposed their conceptual design of the HeLPS simulation system. Lin and Haas [5] continued the work and designed a system being able to perform initial setup planning for cranes and performance measurements for user defined paths. Varghese et al. [6] extended the HeLPS system by monitoring safety factors during interactions. Chadalavada and Varghese [7] developed a plug-in approach for the Autodesk Inventor with their CLPS simulation system. Their solution approach was made up of plant modeling, interactive manipulation and comprehensive safety monitoring.

Sub-problems of lift planning such as crane selection, feasibility checking and crane layout have also been addressed by other researches. Olearczyk et al. [8] discussed the crane selection and locating problem concerning lifting capacities and clearances. The crane location determination problem was solved by optimizing weighted distances from crane locations to pick and place locations of the lifting targets constrained by clearances of tail swing, boom and outriggers. Safouhi et al. [9] and Lei et al. [10,11] dealt with the crane location determination problem using geometric analysis on 2D CAD drawings. In particular, Lei et al. [10] also proposed a method for feasibility checking of multiple lifting cases. The method mapped the pick and place areas into the configuration space (C-space) of the crane and checks the mapped areas with the obstacle regions (C-obstacle). Lei et al. [11] discussed the scenarios where pick and place locations are overly separated and thus a crawler crane is required to walk (or crawl) towards the place location. Similar to Safouhi's idea [9], The method dilated the obstacle regions by the size of the lifting target and the tail-swing radius of the crane. Lei's method was able to provide walking paths of crawler cranes as 2D lines without interference with the dilated obstacles. Sometimes, multiple cranes are required to work together. The algorithm by [12] addressed the multiple tower crane layout problem in construction sites. A hybrid particle bee algorithm was applied to solve the layout problem and better results were reported compared with other types of algorithms.

So far, in all previous studies and existing systems, automatic lifting path planning is attempted mostly at theoretical level with rare implementation reported for practical uses. This is partially because the problem itself is very challenging due to the complexity of plant environments and cranes. The three major concerns of lifting path planning are efficiency, solution quality and success rate. The existing methods used combinations of different search algorithms and collision detection strategies to fulfill the above mentioned criteria. The first class of methods utilized global optimum search algorithms together with C-spaces with precomputed collision information to achieve high solution quality. Sivakumar et al. [13] considered the simplified representation of cranes as planar kinematic chains with two rotational DOFs. Simple Genetic Algorithm (SGA) was performed on the 2D C-space where precomputed collision results were factored in the fitness function as violation penalties. Ali et al. [14] designed a two-stage fitness function and used parameter-based reproduction operators for the Genetic Algorithm (GA) search in a 3D C-space. Both of the GA-based methods were able to achieve highly optimized solutions. However, these methods were computationally forbidding due to the computational intensive nature of GA. Ali's method also suffered from the high computational cost for generating the 3D C-space. As a result, the methods only managed to deal with simple CAD plants. The second class of methods also relied on precomputed collision information. Instead of using the global optimization algorithms, this class of methods use fast search algorithms for finding good but not necessarily optimized collision-free lifting paths. The method by Reddy and Varghese [15] represented cranes as linked rigid bodies with three DOFs (swinging, luffing and hoisting). Heuristic depth first search was performed in the free space. Given a simple CAD plant environment, their planner was able to achieve good solutions as arrays of independent configurations. The algorithm, however, still required the substantial time and memory to generate the 3D free space.

Research by [16] investigated the use of the probabilistic road-map method in dealing with crane erection planning. They proposed a useful idea that, given a 2.5D site with only the maximum height of the plant taken into consideration, a 2D C-space is enough for the computation. Each position of the 2D C-space stores the maximum and minimum hoist height of the crane. Using this 2D C-space, they were able to achieve near real-time solutions in simple environments. Olearczyk et al. [17] tried to conquer the lifting path planning problem by constraining the movement of the lifting target into a single horizontal plane. A* search was conducted in the precomputed 2D ray-arc intersection map. Their algorithm was fast but relatively impractical because of the planar constraint of the position of the lifting target. The algorithms by [16] and [17] could quickly generate a crane lifting path. But the solution quality was somehow compromised compared to the methods using GA. Another class of methods did not rely on the precomputed free space. Instead, collision detection was performed on the fly during the search (which is referred to as "online collision detection" in this paper). This online collision check strategy is less affected by the number of DOFs and thus could be efficiently extended to high-DOF cranes. Kang et al. [18] developed an online collision check based motion planning system using bonding sphere based collision check. The bonding spheres were also used in the continuous collision detection (CCD) which detected interference between the environment and the swept spaces of objects between consecutive movement steps [19]. Bi-directional expanding trees were exploited to search in the Cartesian space for 4-DOF tower cranes. Their algorithm involved three sub-phases: path planning for the end effector (lifting target), crane trajectory coordination and trajectory smoothing. The method could produce collision-free lifting path in short time. But the success rate was restricted because of the overestimated proximity information and the multiple sub-phases. Lin et.al [20] published their impressive work on the crawler crane lifting path planning problem. The crawler crane was regarded as a robot with seven DOFs. The problem was solved by the bidirectional RRT search in a 7D C-space. Their algorithm reported reasonable planning time. The solution quality was good for low-DOF lifting tasks (≤ 3) but produced more zigzagged paths for higher-DOF lifting tasks.

### 1.2. The GPU based MSPGA

Graphics Processing Units (GPUs), as the kernel of graphic hardware, offer tremendous computational horsepower and high memory bandwidth. The computational powers of GPUs have brought significant speedups in various applications [21,22]. Modern General Purpose GPUs (GPGPUs) have the ability of handling general, complex and massive computations. A GPGPU contains several Streaming Multiprocessors (SMs) equipped with caches and control units. It is capable of running hundreds of threads concurrently.

CUDA C/C++ [21] is a popular GPU accessing API designed by nVIDIA. It performs as an extension of the standard C/C++ language [21,23]. CUDA C/C++ provides access to the complete hierarchy of GPU memory, from global memory to register memory inside processors.

Unlike other robotic path planning scenarios which may rely on fast algorithms to achieve reasonable but not necessarily optimal results, crane lifting path planning is eager for global optimization capabilities. Master–slave parallel GA is a class of parallel GAs using a master processor to control the flow and assigning functional components into multiple processors for parallel computing [24]. It is the most popular type of parallel GA for applications due to its predictability and preservation of the global optimization ability of serial SGA. Among the existing researches on GPU accelerated GA, GPU based MSPGA takes up a major part.

Fitness evaluation is usually the most computationally intensive part in GA, especially if computations like proximity computations and collision detections were involved [25]. The GPU implementation of binary-coded and real-coded MSPGA was discussed by [26]. In the algorithm of [27], the GPU based steady-state MSPGA was used as a function optimizer. Wang et al. [28] made use of the GPU based MSPGA in performing daily activity planning. Moreover, researches like Fujimoto et al. [29] emphasized the design of parallel genetic operators.

### 1.3. Outline of our research

The aim of this paper is to fully exploit the global optimization capability of the GAs and the scalability of online collision check strategy. Compared to the serial GA used by [14], this paper utilizes the GPU based MSPGA to handle the computational load of GA searches and online collision detections for complex environment. Furthermore, instead of the iterative CCD used in [18] and [20], the proposed method generates analytical swept spaces according to the features of terrain crane operations which can be computed and checked in parallel. With the collision detection, continuous collision detection and genetic algorithm designed in a massively parallel manner, the overall computation time of the optimized lifting path planning can be significantly reduced. This paper also proposes a hybrid C-space strategy to further improve the efficiency through an innovative combination of the online collision detections with a 2D C-space for the two base DOFs (swinging and luffing) of the 4-DOF terrain cranes (Fig. 1).

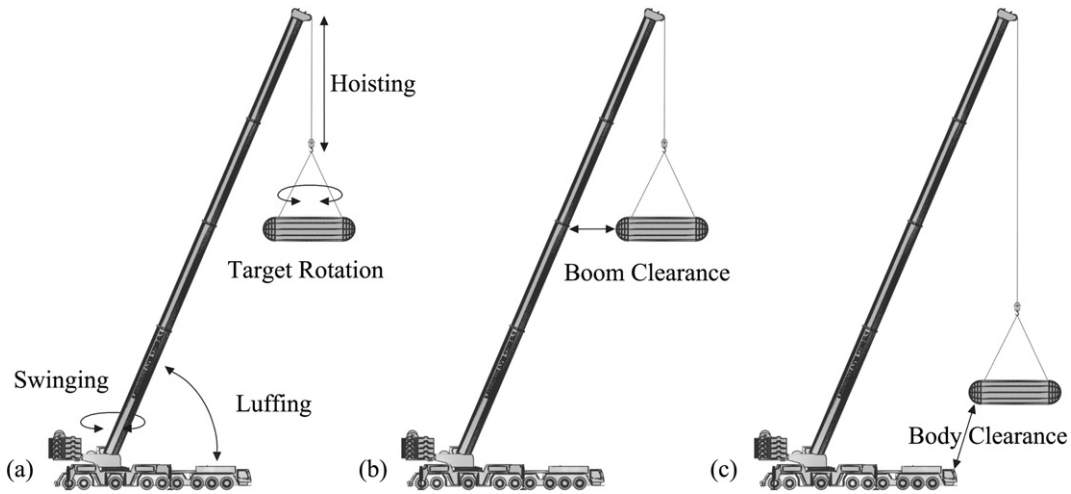



**Fig. 1.** (a) DOFs of the terrain cranes, (b) boom clearance and (c) body clearance.

**Table 1**
Parameters and variables used in solution representation.

| Symbol | Expression |
|---|---|
| $c_i$ | The $i_{th}$ configuration in the string |
| $e_j$ | The $j_{th}$ edge in the string |
| $L_s$ | The length (number of configurations) of the string |
| $\alpha_{LF}$ | The luffing angle in degrees (angle between main boom and ground) |
| $\alpha_{SW}$ | The swinging angle in degrees (rotation angle of main boom along the z axis) |
| $l_{HS}$ | The hoisting length in centimeters (extension length of the sling) |
| $\alpha_{LR}$ | The rotation angle of the load along the z axis |

**Table 2**
Parameters and variables in the objective function.

| Symbol | Expression |
|---|---|
| $F(s)$ | The evaluation value of string $s$ |
| $sc(s)$ | The operation switching cost in string $s$ |
| $d(s)$ | The distance cost in string $s$ |
| $c_j$ | The $j_{th}$ configuration in the string |
| $\lambda_1$ | The constant scaling factor 1 |
| $\lambda_2$ | The constant scaling factor 2 |

**Table 3**
Parameters and variables used in the fitness function design.

| Symbol | Expression |
|---|---|
| $f(s_i)$ | The fitness value of string $s_i$ |
| $s_i$ | The $i$th string in the population |
| $n_i$ | The collision violation number |
| $m_i$ | The motional cost |
| $L_p$ | The size of population |
| $no_i$ | The collision violation number of Oriented Bounding Boxes (OBBs) in string $s_i$ |
| $nf_i$ | The collision violation number of boom swept volumes in string $s_i$ |
| $nr_i$ | The collision violation number of load swept volumes in string $s_i$ |
| $nc_i$ | The collision violation number of internal clearance in string $s_i$ |
| $sc_i$ | The operation switching count in string $s_i$ |

## 2. Problem formulation

The goal of the crane lifting path planning problem is to minimize the costs during lifting operations. These costs include the time and energy consumption of the crane, safety risks during the operation and the difficulty for human operators to follow the lifting path. We develop a mathematical model for the lifting problems. The solution space, objective function and constraints of the lifting path planning problem are analyzed in this section.

### 2.1. Assumptions

The mathematical model of the problem is established upon the following observations and assumptions:

(1) The terrain crane is not allowed to drive during lifting.
(2) Booms of the terrain cranes are strictly not allowed to extend or retract during lifting processes.
(3) The lifting target can be rotated (manually by the rigging man) near the start or end positions.
(4) It is not permitted to perform the three classes of operations below simultaneously: boom swinging, boom luffing and sling extension, and target rotation.
(5) The speeds of elementary operations are constant (usually very low) and the corresponding energy cost of the crane is proportional to the movement units.
(6) Components of the cranes and the lifting targets are not supposed to be operated below any plant structures.

### 2.2. Mathematical formulation

According to observations 1, 2 and 3, a terrain crane has four DOFs during lifting operations: boom swinging, boom luffing, hoisting (sling extension and shortening) and target rotation (restrained in start and end positions) (Fig. 1(a)). Parameters of the DOFs are constrained into limits determined by the internal clearances including boom clearance and body clearance (Fig. 1(b)). The entire set of parameters specifying the four DOFs form a configuration of the crane. The set of all possible configurations is defined as the C-space (denoted as $C$ in this paper), which can be represented as:

$$C = SO(3) \times \mathbb{R} \times SO(2).$$

Here $SO(3)$ represents the group of rotational matrices that define the two rotational DOFs (swinging and luffing) of the main boom and $\mathbb{R}$ related to the translational motion of the sling. The last item $SO(2)$ represents the rotational DOF of the lifting target along the vertical axis.

Typically, a crane lifting path is defined as an array of configurations or steps. This configuration array stands for a general polyline in $C$. However, this definition of lifting path is not going to assure assumption 4. According to assumption 4, some portions of the polyline need to be axis-aligned. This constraint requires the GA to handle inter-dependent genes (configurations) which arise difficulties in designing crossover and mutation operators.

Thus we propose to formulate the lifting path as a structure which provides independent configurations for GA search and remains as axis-aligned polylines for processing (such as collision detection) in the meantime.

In the proposed algorithm, the lifting path is represented as a string $s = \{O, E\}$, where $O$ is the set of nodes (configurations) and $E$ represents the set of edges (internal paths between independent nodes). Thus the variables of the optimization problem can be written as (see in Table 1 for explanations of the symbols):

$$\begin{aligned} s &= \{c_i\}_{i=0,1,\ldots,L_s-1} \cup \{e_j\}_{j=0,1,\ldots,L_s-2} \\ c_i &= (\alpha_{LF}, \alpha_{SW}, l_{HS}, \alpha_{LR}). \end{aligned} \quad (1)$$

The edges $e_j$ are composed by set of key frame configurations determined by its neighboring nodes $c_j$ and $c_{j+1}$ in a predefined way which

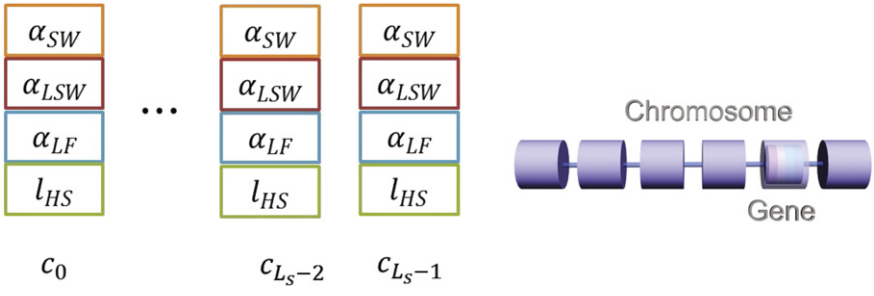



**Fig. 2.** Encoding structure of chromosomes.

will be discussed in Section 3.3. This definition of inputs will be reorganized as chromosomes in the GA search in Section 1.2. The task of the planning algorithm is to find an optimal $s^*$ composed of $c_j^*$ which maximizes the evaluation function.

In order to design an evaluation function for the strings, we define metric functions in $C$, so that $C$ becomes a metric space. Those two metrics, $d_1$ and $d_2$, can be defined as:

$$d_1(a,b)=\sum_{i=0}^{3} r_i|a_i-b_i|; \quad d_2(a,b)=\sum_{i=0}^{3} g(a_i-b_i).$$

Here $a$ and $b$ denote two configurations in space $C$. $a_i$ and $b_i (i = 0, \ldots, 3)$ are the unified representation of the four parameters of $a$ and $b$ respectively (Eq. 1). Note that a scaling factor $r_i$ is applied to the absolute difference value for each dimension in $d_1$. Function $g$ in $d_2$ is defined as:

$$g(x)=\begin{cases}1 & \text{if } x\neq 0, x\in R\\ 0 & \text{if } x=0, x\in R\end{cases}.$$

Here $d_1$ measures the total number of movement units along the four dimensions (weighted). $d_2$ represents the number of non-identical parameters between the two configurations. Now the evaluation function of a string $s$ in the solution space $S$ can be expressed as (see in Table 2 for explanations of the symbols):

$$F(s)=\lambda_1\left(1+\frac{\lambda_1}{d(s)+\lambda_2(1+sc(s))}\right)$$

*where*

$$d(s)=\sum_{i=0}^{L_s-2} d_1(c_i,c_{i+1})$$
$$sc(s)=\sum_{i=0}^{L_s-2} d_2(c_i,c_{i+1})$$
$$s\in S, c_i\in C \quad \text{and} \quad i=0,\ldots,L_s-1.$$

**Table 4**
Parameters and variables in adaptive mutation rates.

| Symbol | Expression |
|---|---|
| $r_m(s)$ | The mutation rate of string s |
| $\overline{r_m}$ | The basic mutation rate |
| $\overline{f}$ | The average fitness value in the population |
| $f(s)$ | The fitness value of string s |

**Table 5**
The initialization strategy.

| Configuration | Strategy |
|---|---|
| $c_1$ | Randomly generate sling length; Keep other parameters as the start configuration |
| $c_2 \sim c_{L_s-3}$ | Randomly generate parameters |
| $c_{L_s-2}$ | Randomly generate sling length; Keep other parameters as the end configuration |

Then the maximizing optimization problem for the lifting path planning scenario can be accordingly written as:

$$\begin{aligned}
&\max \quad F(s)\\
&s.t. \quad n_{node}(s)=0, \ \ s\in S\\
&\qquad n_{edge}(s)=0, \ \ s\in S\\
&\qquad cl(s)=0, \ \ s\in S\\
&\qquad \underline{B}\le c_i\le \overline{B},\\
&\qquad i=0,1,\ldots,L_s-1\\
&where \quad n_{node}(s)=\sum_{i=0}^{L_s-1}\delta(c_i)\\
&\qquad n_{edge}(s)=\sum_{i=0}^{L_s-2}\delta(e_i)\\
&\qquad cl(s)=\sum_{i=0}^{L_s-1}\sigma(c_i)\\
&\qquad \delta(c_i)\in\{0,1\}, i=0,1,2\ldots,L_s-1\\
&\qquad \delta(e_i)\in\{0,1\}, i=0,1,2\ldots,L_s-2\\
&\qquad \sigma(c_i)\in\{0,1\}, i=0,1,2\ldots,L_s-1.
\end{aligned} \tag{2}$$

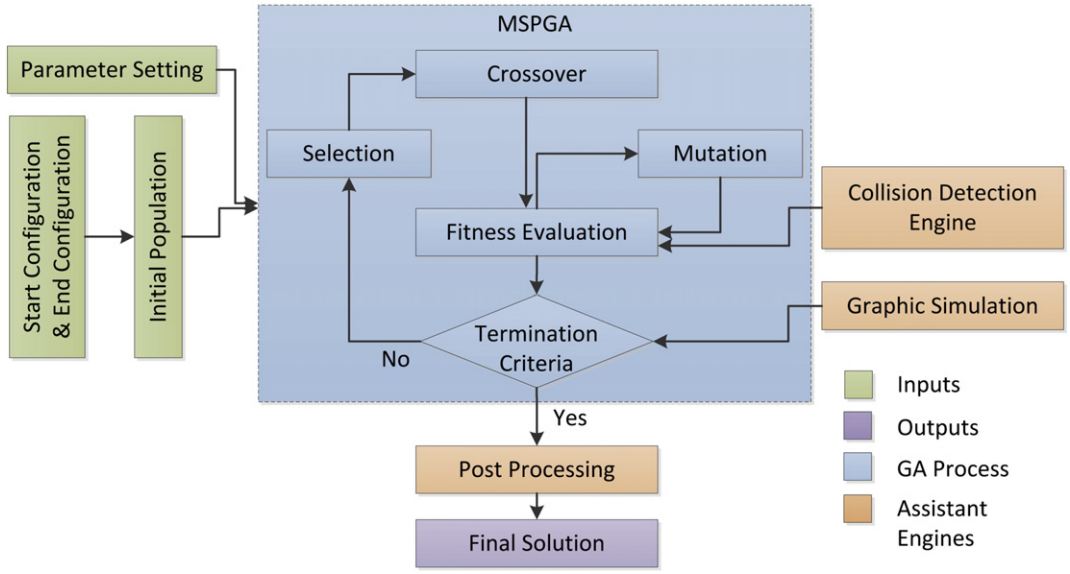



**Fig. 3.** Framework of the lifting path planning system.

**Table 6**
The crossover strategy.

| Configuration | Parents | Strategy |
|---|---|---|
| $c_1$ | Both valid | Choose shorter sling length |
| | Both invalid | Random choose |
| | One valid, one invalid | Choose valid |
| $c_2 \sim c_{L_s-3}$ | Both valid | Random choose |
| | Both invalid | Random choose |
| | One valid, one invalid | Choose valid |
| $c_{L_s-2}$ | Both valid | Choose longer sling length |
| | Both invalid | Random choose |
| | One valid, one invalid | Choose valid |

**Table 7**
The mutation strategy.

| Configuration | Case | Strategy |
|---|---|---|
| $c_1 \& c_{L_s-2}$ | Valid | Randomly alter sling length in smaller scale |
| | Invalid | Randomly alter sling length in larger scale |
| $c_2 \sim c_{L_s-3}$ | Valid | Randomly alter all parameters in smaller scale |
| | Invalid | Randomly alter all parameters in larger scale |

Here $B$ and $B$ stand for the lower and upper bound values for the configurations. $\delta(c_i)$ and $\delta(e_i)$ represent the collision detection results of elements $c_i$ and $e_i$ in string $s$. $\sigma(c_i)$ stands for internal clearance checking result for configuration $c_i$. Accordingly, $n_{node}(s)$, $n_{edge}(s)$ and $cl(s)$ represent the collision violation factors of node configurations, edge paths and internal clearance within the crane itself. These values are calculated from the complex geometric information in the Euclidean space.

Feasible solutions of the maximization problem contain no violation for collision and inter-collision. The motion cost $d(s)$ and operation switching cost $sc(s)$ are minimized through maximizing function $F(s)$, making the optimal solution $s^*$ short in distance and comfortable for human operators. The scale factors enlarge the difference between good solutions and bad solutions, which helps the convergence of the GA. As a result, the optimal solution $s^*$ of the maximization problem is a collision-free lifting path which is optimized in energy cost and human operation conformity.

The mathematical and algorithmic details of the computations will be discussed in Section 4.

### 2.3. Fitness function

When transferring the optimization problem into the language of GA, we add the hard constraints of the optimization problem (Eq. 2) into the fitness function as penalties. The fitness function for a given chromosome $s_i$ in the population $P$ is defined as (see in Table 3 for explanations of the symbols):

$$f(s_i) = \begin{cases} \lambda_1 / n_i & \text{if } n_i > 0 \\ \lambda_1 (1 + \lambda_1 / m_i) & \text{if } n_i = 0 \end{cases} \quad (3)$$

*where*

$$n_i = no_i + nf_i + nr_i + nc_i$$
$$m_i = d_i + \lambda_2 (1 + sc_i)$$
$$i = 0, 1, 2, \ldots, L_p - 1.$$

When $n_i > 0$, some of the nodes or edges in string $s_i$ are colliding with the environment or the crane itself. In such case, the fitness function focuses on the elimination of collisions. Once $n_i = 0$, it means that $s_i$ becomes a feasible solution. The functionality of GA turns into optimization of the objective function in the feasible space. This fitness function is an extended version of the one utilized in our previous work [30]. It includes the internal clearance as a hard constraint and counts the number of operation switching into the motion cost.

## 3. GPU-based MSPGA solver for the lifting path planning problem

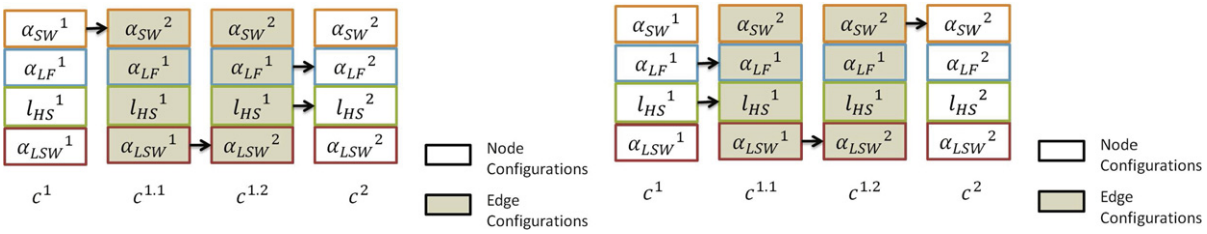
As a multi-objective non-linear integer optimization model, the lifting path planning problem is challenging for common combinatorial optimization methods. GA, as a general optimization algorithm which is mathematically proved to be able to achieve global optimum, is highly suitable for the task. It has good potentials for customization and parallelization. Our previous work [30] presented a GPU based MSPGA framework (which we will briefly introduce in Section 3.1 to make this paper self-contained). GPU parallelization for the MSPGA framework can be found in [30] and a simple lifting path planning problem was investigated to test its efficiency and scalability. In this paper we will present an in-depth discussion on designing the customized adaptive plan for the lifting path planning problem in complex environments. A post processing stage is also introduced to improve the human conformity of the result path. Based on the framework, we investigate the use of multi-layered depth maps to deal with discrete and continuous collision detection in complex environments and propose a hybrid C-space collision strategy for improving the efficiency of the collision detection module. The collision detection module will be further discussed in Section 4.

A chromosome (Fig. 2) in our MSPGA framework is defined as the array of configurations (genes) taking all the node configurations $c_i (i = 0, \ldots, L_s - 1)$ from the solution $s$ in Eq. 1. The population is thus a set of chromosomes carrying different path candidates evolving in the GA process.

### 3.1. MSPGA framework

The functioning of the MSPGA relies on a complete system with software and hardware components. The software components provide a simulation environment for generating necessary inputs (crane position, boom length, destination configurations, etc.) and displaying outputs (paths, costs, capacities etc.). Fig. 3 is a brief denotation of the system.

The master processor we use for the MSPGA is the CPU. The four functional components, fitness evaluation, selection, crossover and mutation, are handled by the GPU. In each iteration of GA, GPU kernels for




(a) When the target position in $c^1$ is higher; (b) when the target position in $c^2$ is higher

**Fig. 4.** The post processing strategy.

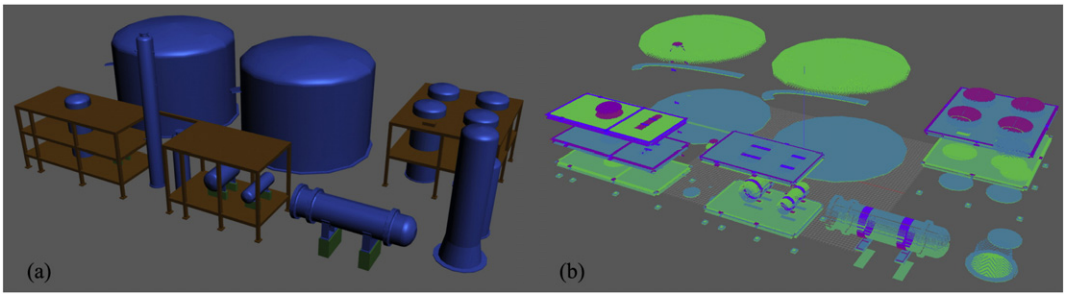
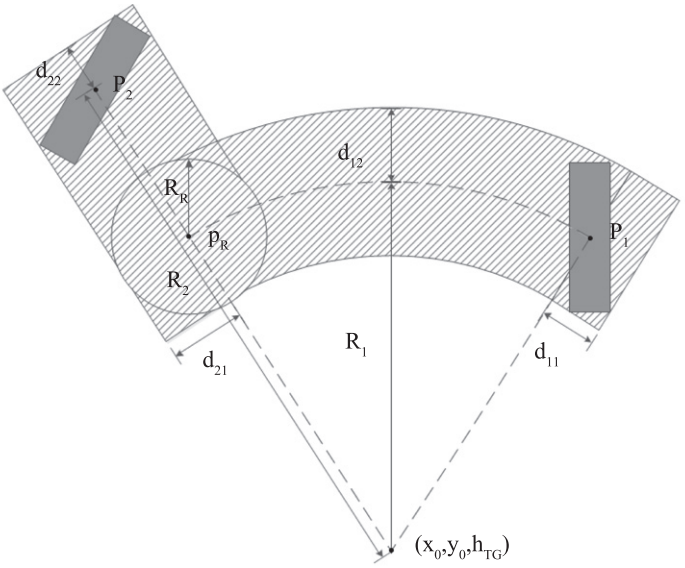
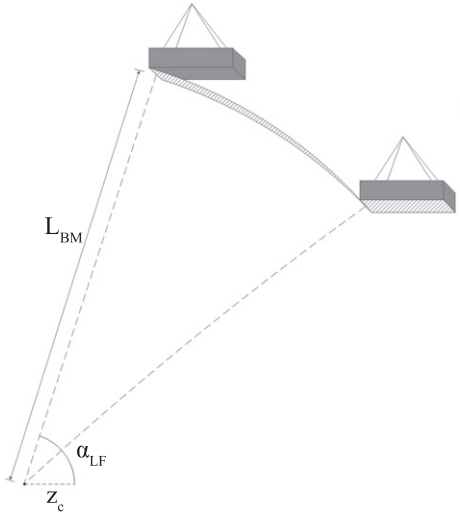



**Fig. 5.** Digitization of the plant model using depth map: (a) Original scene, (b) Multi-layered depth map. Different colors stands for layers in the multi-layered depth map.

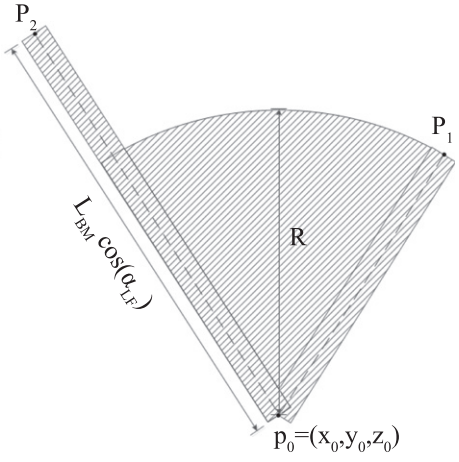
the four components are executed in sequence. Among these components, selection and crossover select good candidates from the population to produce offspring. The mutation process randomly alters genes in offspring which help GA to find better configurations in the neighboring spaces. At the end of each iteration, fitness values of the new population are returned to the CPU to be checked against the termination fitness value. Once the termination fitness is met, the CPU stops the GA process and extracts the optimum chromosome from the GPU memory. If the search exceeds a certain number of iterations, the CPU will stop the search and report failure. As MSPGA preserves the




(a) Target swept frontier from top view




(b) Target swept frontier of luffing from side view




(c) Boom swept frontier from top view

**Fig. 6.** Demonstration of the swept frontier of the swinging and luffing operations.

property of probabilistic completeness of SGA, the possibility of failure will decrease to zero when the number of iterations increases.

It is difficult to find a set of termination conditions which guarantee a fully optimized result for a randomized search algorithm like GA. Thus, apart from the basic termination criteria stated above, we seek additional help from the simulation environment. The final solution, or lifting path, with its properties is displayed as animations for real-time monitoring. Based on the graphical results, users can choose whether to conduct further searches.

### 3.2. Adaptive plan

Initialization of the population is the basis of GA. It provides the initial resources and information for the GA to start the search. Our initialization uses the strategy shown in Table 5. $c_0$ and $c_{L_s-1}$ in the strings are the start and end configurations of the task. For $c_1$ and $c_{L_s-2}$, the hoisting lengths are randomly generated and other values are kept as in the start (for $c_1$) or the end configuration (for $c_{L_s-2}$). Internal configurations are generated randomly within the bound values.

Selection, crossover and mutation are referred to as the reproductive operators in GA. They are the key components of an adaptive plan which are performed in the evolutionary iterations. By designing proper reproductive operators, the GA can achieve higher convergence speed and, in the meanwhile, produce high quality solutions.

The selection operator reflects the concept of "survival of the fittest" in Darwinian evolution. A good selection operator provides better chance for "fitter" individual to survive and reproduce. In our algorithm, a proportional selection scheme is used together with the elitism strategy. Namely, the "fittest" chromosome in the population always survives and remains in the next generation. For other chromosomes, the chances of producing off-springs are proportional to their fitness values.

Crossover happens with a given rate $r_c$ when parent strings are mating. Its mathematical essence is to direct the search to "fitter" areas in the solution space by combining information in existing solutions. In the proposed method, crossover is also responsible for eliminating invalid (colliding) configurations from the population. Our approach exploits a parameter based crossover strategy which is illustrated in Table 6. For $c_1$ and $c_{L_s-2}$, the off-springs inherit the higher target positions between their parents. In-between configurations inherit valid configurations from the parents in the sense of collision avoidance and internal clearance. The purpose of this strategy is to help GA enter the feasible (collision-free) space through giving higher priority to genes with better potential for collision avoidance.

The mutation operator alters bits (genes) in existing chromosomes with a given rate $r_m$. Chromosomes with lower fitness values can thus perform as seeds for exploring unknown areas in the solution space. Larger mutation rates would help the GA in finding new possibilities, but at the same time increase the likelihood of damaging existing good chromosomes. On the other hand, lower mutation rates help to preserve known solutions but slow down the convergence.

This analysis leads to adaptive mutation rates. In the proposed approach, the mutation rates for chromosomes are formulated as (see in Table 4 for the symbols):

$$r_m(s) = \begin{cases} \overline{r_m} + \left(\overline{f} - f(s)\right)/\overline{f}, & \text{if } f(s) < \overline{f} \\ \overline{r_m}, & \text{if } f(s) \geq \overline{f} \end{cases}$$

The mutation strategy used in the proposed algorithm is denoted in Table 7. Note that for $c_1$ and $c_{L_s-2}$ in each chromosome, the mutation only alters the hoisting length (sling length). This design comes from the observation that the first and last step of lifting operations are always hoisting or lowering the target.

### 3.3. Post processing

The task of post processing is to build configurations in the edges from the node configurations returned by the GA search. As we have mentioned in Section 2, the cranes are not allowed to perform the three classes of operations simultaneously. Thus an edge $e$ also contains three segments: boom swinging, luffing, hoisting and load rotation. With the movement units already provided by the node configurations, the key of the edge building strategy is to define the sequence of conducting the three classes of operations.

From our observation, when conducting boom swinging, it would be safer for the target to be in higher positions than in lower ones. Accordingly, the order of boom swinging and hoisting for a given edge $e$ should be decided by the target height in the two neighboring node configurations $c^1$ (left neighbor) and $c^2$ (right neighbor). For example, if the target is at higher position in the configuration $c^1$, then boom swinging should be conducted ahead of hoisting. Rotation of the target is located in-between the other two types of operations. Fig. 4 indicates the stated strategy of edge building which identifies the key frame configurations ($c^1$, $c^{1.1}$, $c^{1.2}$, $c^2$) for edge $e$.

This strategy is also applied when performing the continuous collision detection for edge paths (Section 4.2). In this way the solution provided by the GA search will be guaranteed to be valid after the post-processing.

## 4. Collision avoidance

The proposed algorithm does not rely on precomputed collision information in the C-space. The GA, therefore, needs to perform online collision detection when running its iterations. For collision detection of the nodes, an image-space parallel collision detection algorithm is designed and implemented in GPU, which is initially discussed in Cai et al. [22]. In this paper we present a complete and practical implementation of the algorithm. For collision avoidance of the edges, we propose a CCD method using analytical representations of the swept volumes of the crane.

**Table 8**
Parameters and variables in the swept frontiers.

| Symbol | Expression |
|---|---|
| $A_{TG}$ | The swept frontier for the lifting target |
| $A_{TG\_SW}$ | The swept frontier for the lifting target during swinging |
| $A_{TG\_LF}$ | The swept frontier for the lifting target during luffing |
| $A_{TG\_LR}$ | The swept frontier for the lifting target during target rotation |
| $A_{BM}$ | The swept frontier of booms |
| $p$ | A point on the swept frontier |
| $p_0$ | The center of rotation of the main boom |
| $x_0$ | The x coordinate value of $p_0$ |
| $y_0$ | The y coordinate value of $p_0$ |
| $z_0$ | The z coordinate value of $p_0$ |
| $z_c$ | The height of the virtual center of rotation of the lifting target during luffing |
| $p_1$ | The location of load center at the first swinging angle |
| $p_2$ | The location of load center at the second swinging angle |
| $p_R$ | The location at which load rotation happens |
| $\alpha_{LF}$ | The constant luffing angle during swinging in the internal path |
| $\alpha_{SW}^1$ | The swinging angle in gene 1 |
| $\alpha_{SW}^2$ | The swinging angle in gene 2 |
| $d_{11}$ | The width of the lifting target at the tangential direction in gene 1 |
| $d_{12}$ | The width of the lifting target at radial direction in gene 1 |
| $d_{21}$ | The width of the lifting target at the tangential direction in gene 2 |
| $d_{22}$ | The width of the lifting target at radial direction in gene 2 |
| $R_1$ | The working radius of the crane at gene1 |
| $R_2$ | The working radius of the crane at gene2 |
| $R_R$ | The working radius at which load rotation happens |
| $h_{TG}$ | The height of the target during swinging |
| $L_{BM}$ | The total length of the boom |

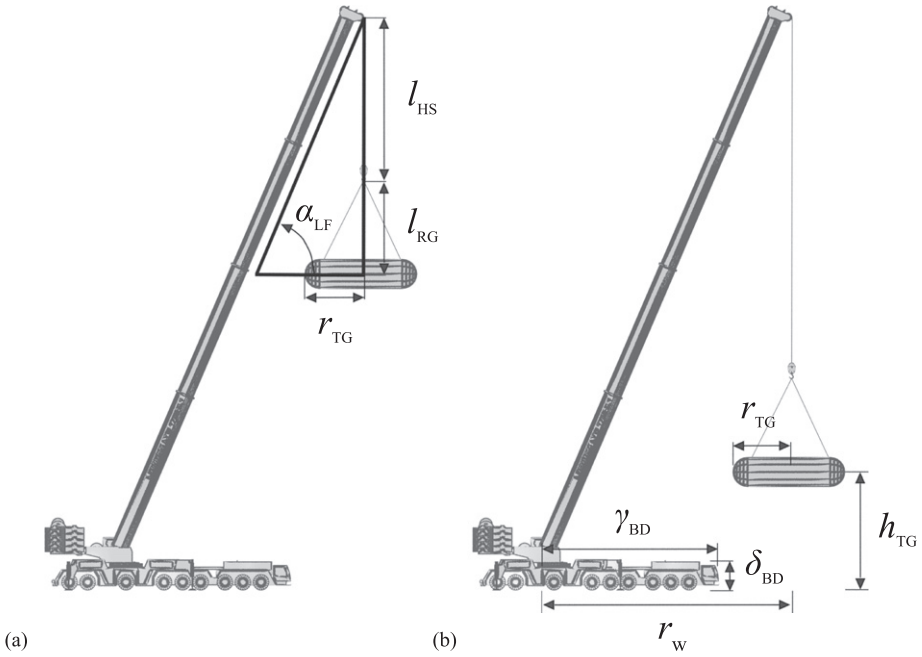



**Fig. 7.** Demonstration of variables used in the computation of internal clearance for terrain cranes. (a) Load-boom clearance, (b) load-body clearance.

### 4.1. Discrete collision detection

Petrochemical and pharmaceutical plants usually have highly complicated structures. According to assumption 6 in Section 2, terrain cranes are not supposed to move or operate below any plant structures. Thus, for a crane, the frontier of contact with the plant structure is decided by the plant structure's highest portions. As such, we can transfer the plant model into a histogram shaped structure by simply computing a depth map of the model. This representation will be used in both the Discrete Collision Detection (DCD) and continuous collision detection algorithms. Benefiting from the highly ordered feature of the depth maps, the collision detection can be parallelized in three levels: pixel level, gene level and chromosome level. This three-level parallelization enables the high efficiency of our collision detection algorithm.

The procedure of the discrete collision detection algorithm would be:

(1) Computing an initial oriented bounding box (OBB) hierarchy for the crane model
(2) Generating the depth map for the plant model
(3) Updating the OBB hierarchy for the crane model before each collision check in step 4, and
(4) Performing collision check between the crane OBB hierarchy and the plant depth map

Steps 1 and 2 in the flow are conducted in the initialization stage while Step 3 and 4 are done in two places. For simulation purpose, it is performed as runtime collision detection. For the GA search, it serves as a component in the fitness evaluation process. Fig. 5 shows the result of the generated depth map compared with the original triangular model.

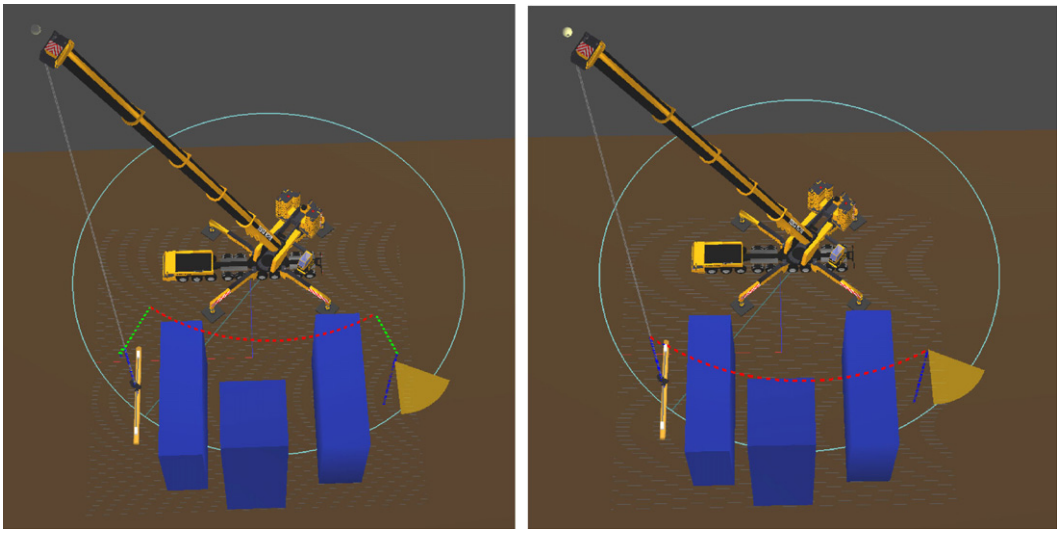
For fast generation of a GPU depth map from the original model, we borrow the concept of OpenGL rasterization and design a GPU depth map generator. In this GPU depth map generator, each GPU thread block takes care of one triangle face passed from the host memory. Parallel threads in GPU thread blocks are mapped into the x–y ranges of the triangle faces. The threads cast vertical rays from the ground and the intersection positions with the triangle faces are stored in the corresponding positions in the depth map. Some pixels in the depth map may have intersections with more than one triangle face. For generation of single-layer depth map which only takes the highest intersection, the system uses atomic functions [21] to prevent conflict in memory locations. The collision detection can be scaled up from 2.5D to 3D by recording a multi-layer depth map which is indicated in Fig. 5(b). Generation of the 3D depth map requires a precomputed map recording the number of intersections for each pixel.

**Table 9**
Parameters and variables in the internal clearance inequalities.

| Symbol | Expression |
|---|---|
| $\alpha_{LF}$ | The main boom angle (luffing angle) |
| $l_{HS}$ | The length of the sling (hoisting length) |
| $l_{RG}$ | The distance between bottom tip of the hook and center of the target |
| $r_{TG}$ | The radius of the bounding sphere of the target |
| $h_{TG}$ | The height of the bottom face of the target |
| $\delta_{BD}$ | The height of the crane body |
| $r_w$ | The working radius of the crane |
| $\gamma_{BD}$ | The distance between head of the crane body and the rotational axis of booms |

**Table 10**
Inputs into Experiment 1.1.

| Input | Value |
|---|---|
| Crane model Terex | AC700 |
| Size of population | 100 |
| Length of string | 6 |
| Crossover rate ($r_c$) | 0.15 |
| Mutation rate ($r_m$) | 0.75 |
| Scale of mutation 1 | 0.016 |
| Scale of mutation 2 | 0.16 |
| Boom length (m) | 62.4 |
| Start configuration | (67,119,4972,119) |
| End configuration | (67,52,4972,52) |
| Termination iteration | 400 |



(a) Result path generated using Ali's fitness function: extra luffing (as shown in green colored lines) required.

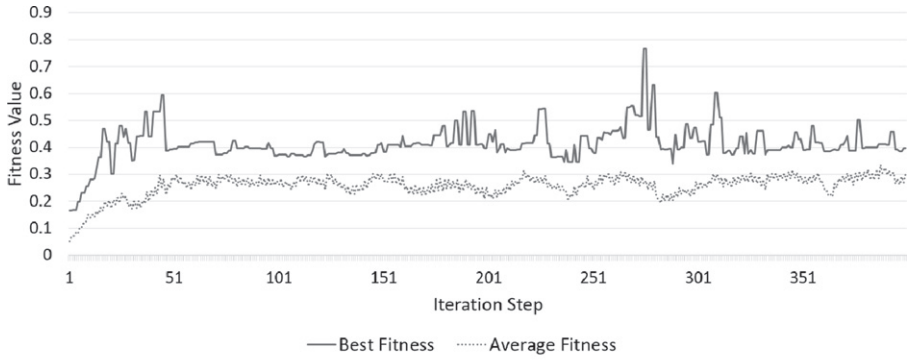
(b) Result path generated using the proposed fitness function: no luffing required.

**Fig. 8.** Result path generated using different fitness functions in Experiment 1.1.

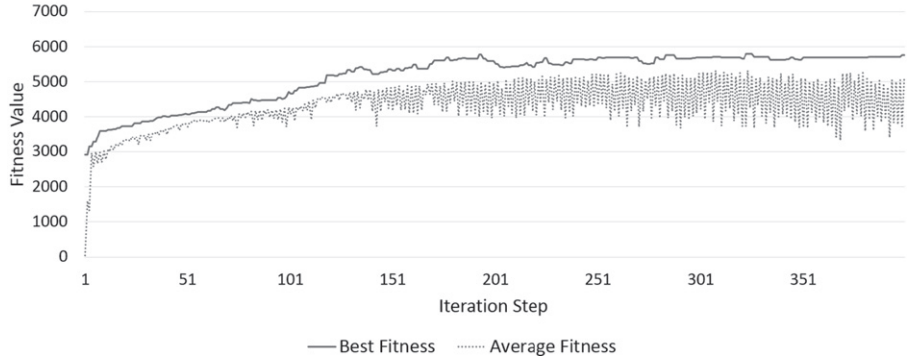
The OBB hierarchy of a terrain crane contains five OBBs. They are OBBs of body, cockpit, counterweight, boom and load, respectively. Bottom portions of these OBBs are the contact frontiers to the plant depth map. As the hook and slings are always above the load for terrain cranes, they are not going to affect the contact frontier. During the GA search, each gene possesses one such set of OBBs. In each iteration, the set of OBBs are updated in the GPU in parallel for the whole population. In the stage of fitness evaluation, another GPU kernel is launched to compute the contact frontiers for the OBBs in parallel and compares the OBB frontiers with points in the depth map.

### 4.2. Continuous collision detection

In order to enable sparsely sampled paths and reduce the number of steps in the lifting path, collision detections for the internal trajectories

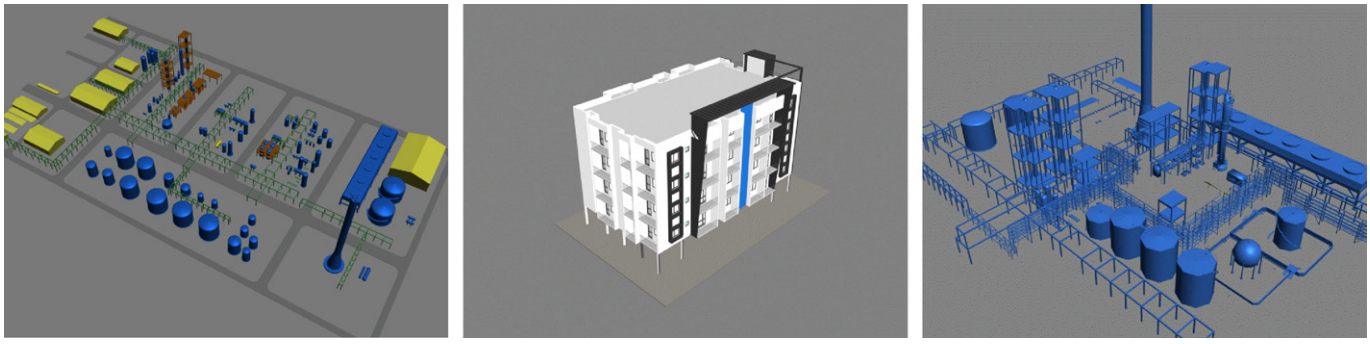



(a) Fitness convergence trend using Ali's fitness function




(b) Fitness convergence trend using the proposed fitness function

**Fig. 9.** Fitness convergence trend using different fitness functions in Experiment 1.1.



(a) Plant 1 (b) Plant 2 (c) Plant 3

**Fig. 10.** Additional plants used in Experiment 1.2.

between consecutive configurations become crucial. This goal is achieved by exploiting the continuous collision detection technology. We introduce the analytical estimations of the swept volumes between consecutive configurations.

During the movements along the internal paths between neighboring genes, the booms and the lifting target have the highest possibility to collide with the plant structure. When the crane moves from one configuration to another, the spaces that the booms and the load swept through are called swept volumes. Bottom faces of these swept volumes are denoted as swept frontiers which are used to perform contact check with the plant depth map. By applying the predefined movement strategy (Section 3.3), we can always get a unique swept frontier for each pair of neighboring genes. Fig. 6 shows the 2D illustrations of the swept frontiers. In the illustrated case, the crane first performs swinging in the counter-clockwise direction, then rotate the target and lower down the boom to reach the destination working radius. Analytically, the swept frontiers for this case are represented as (see in Table 8 for explanations of the symbols):

$$
\begin{aligned}
A_{TG} &= A_{TG\_SW} \cup A_{TG\_LF} \cup A_{TG\_LR} \\
A_{TG\_SW} &= \left\{\left\{d_{xy}(p, \overline{p_0 p_1}) \le d_{11}\right\} \cup \left\{d_{xy}(p, \overline{p_0 p_2}) \le d_{11}\right\} \cup \left\{a_{SW}^1 \le a(p_0 p) \le a_{SW}^2\right\}\right\} \\
&\quad \cap \left\{R_1 - d_{12} \le p_{xy}(p - p_0) \le R_1 + d_{12}\right\} \cap \left\{z = h_{TG}\right\} \\
A_{TG\_LF} &= \left\{d_{xy}(p, \overline{p_0 p_2}) \le d_{21}\right\} \cap \left\{R_2 - d_{22} \le p_{xy}(p - p_0) \le R_2 + d_{22}\right\} \\
&\quad \cap \left\{z = z_c + \left(L_{BM}^2 - (x - x_0)^2 - (y - y_0)^2\right)^{\frac{1}{2}}\right\} \\
A_{TG\_LR} &= \left\{p_{xy}(p - p_R) \le R_R\right\} \cap \left\{z = h_{TG}\right\} \\
A_{BM} &= \left\{\|p - p_0\| L_{BM}\right\} \cap \left\{a_{SW}^1 \le a(\overline{p_0 p}) \le a_{SW}^2\right\} \\
&\quad \cap \left\{z = z_0 + \tan(a_{LF})\left((x - x_0)^2 + (y - y_0)^2\right)\right\}.
\end{aligned}
$$

Here function $d_{xy}$ gets the distance between a point (the first parameter) and a line (the second parameter). Function $p_{xy}$ represents the projection length of a vector on to the $x$–$y$ plane. Function $a$ calculates the planar angle between a given vector and the $x$ axis. In each loop of fitness evaluation, these swept frontiers are compared with points in the depth map of the plant model. The Boolean CCD results serve as parts of the fitness function (Eq. 3).

**Table 11**
Comparison of the success rates in the three plants using the fitness function in [14] and the proposed fitness function in Experiment 1.2.

| | Success rate | |
|---|---|---|
| | Ali's fitness function | The proposed fitness function |
| Plant 1 | 8% | 100% |
| Plant 2 | 88% | 100% |
| Plant 3 | 0% | 96% |

### 4.3. Self-collision clearance

Apart from collision between the crane and environment objects, internal collisions, especially interferences between the lifting target and crane components, are also crucial.

We consider two types of possible internal contacts (Fig. 7) (see in Table 9 for the explanations of symbols). The first type is the clearance between the target and boom segments. This type of clearance check becomes more important when the sling length gets shorter. The mathematical representation would be:

$$cot(\alpha_{LF})(l_{HS} + l_{RG}) > r_{TG}$$

**Table 12**
Solution qualities in the three plants using the fitness function in [14] and the proposed fitness function in Experiment 1.2.

| | Solution quality | | | | |
|---|---|---|---|---|---|
| | | | Plant 1 | Plant 2 | Plant 3 |
| Alis fitness function | Motion units | Swinging (degree) | 93 | 131 | – |
| | | Luffing (degree) | 28 | 25 | – |
| | | Hoisting (m) | 12.06 | 21.43 | – |
| | | Target Rotation (degree) | 93 | 131 | – |
| | Operation steps | | 11 | 12 | – |
| The proposed fitness function | Motion units | Swinging (degree) | 93 | 131 | 83 |
| | | Luffing (degree) | 0 | 3 | 29 |
| | | Hoisting (m) | 22.66 | 21.51 | 49.52 |
| | | Target Rotation (degree) | 93 | 131 | 8 |
| | Operation steps | | 8 | 9 | 11 |

**Table 13**
Inputs into Experiment 2.

| Input | Value |
|---|---|
| Crane model Terex | AC700 |
| Size of population | 100 |
| Length of string | 6 |
| Crossover rate ($r_c$) | 0.15 |
| Mutation rate ($r_m$) | 0.75 |
| Scale of mutation 1 | 0.016 |
| Scale of mutation 2 | 0.16 |
| Weight of boon swinging | 1.0 |
| Weight of boom luffing | 1.5 |
| Weight of sling extension | 0.06 |
| Weight of target rotation | 1.0 |
| Boom length (m) | 62.4 |
| Start configuration | (55,349,4212,349) |
| End configuration | (52,72,4001,341) |
| Termination iteration | 200 |

**Table 14**
Results of Experiment 2.

| Strategy | Preprocessing time (ms) | Planning time (ms) | Success rate | Average success fitness |
|---|---|---|---|---|
| C-space | 10,535 | 789.33 | 66% | 3611.06 |
| Online | 0 | 3998.75 | 93% | 3943.66 |
| Hybrid | 1034 | 2133.79 | 92% | 3965.83 |

The second type of internal clearance is the clearance between the target and the crane body. This type of check is significant when the target is moving in lower positions. The mathematical representation can be written as:

$$h_{TG} > \delta_{BD}, \text{ if } r_w \le \gamma_{BD} + r_{TG}$$

### 4.4. Hybrid C-space strategy

The analytical swept frontier based CCD proposed in Section 4.2 has good potential scalability in terms of the DOFs of the cranes. However, it will also result in longer planning time compared with the C-space based approaches. For cranes with four DOFs, the 2D C-space with minimum-maximum hoisting values proposed by [16] can be a good solution for reducing both the preprocessing time and the planning time. But this approach ignored the fourth DOF of the crane (load rotation). Thus it results in over-estimated proximity information in the 2D C-space. For lifting cases where the crane has to rotate the load into a certain angle to avoid the obstacles, this approach will have low success rates.

We propose a hybrid C-space strategy to balance the preprocessing time and the planning time. In this strategy, collision information for the first two DOFs of the crane (swinging and luffing) are stored in a 2D C-space while the collision check (DCD and CCD) for the last two DOFs (hoisting and load rotation) are conducted during the MSPGA search. Internal clearance check is also conducted during the search. Given the simplicity of the 2D C-space, the hybrid strategy would be able to quickly obtain the C-space and in the meantime to reduce the planning time.

## 5. Results and analysis

### 5.1. Comparison on the fitness function

To evaluate the fitness function proposed in our method, we compare it with the fitness function used in Ali's work [14]. Their algorithm is initially targeting at the dual-crane erection problem, but the fitness function can be easily adapted to the single crane lifting problem. By eliminating the coordination violation coefficient in their fitness function (which is not necessary in the single crane case), we obtain a single-crane version of Ali's fitness function:

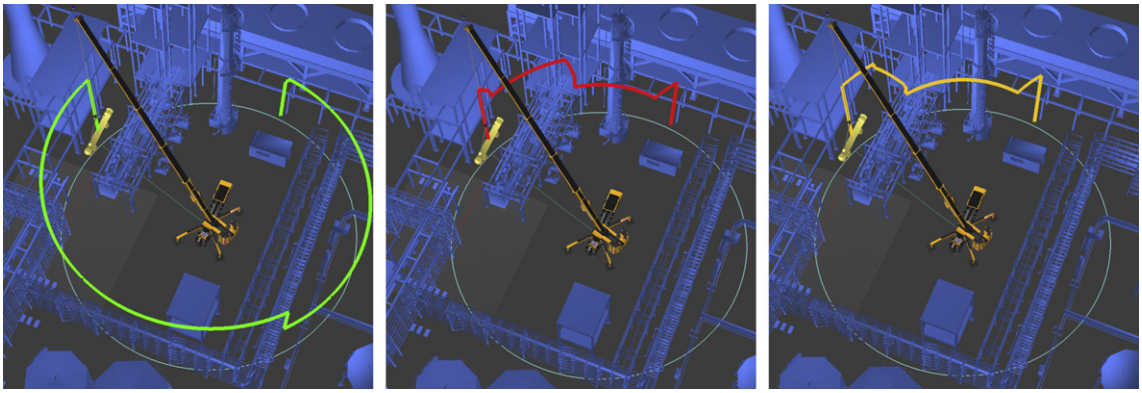
$$\begin{aligned} F(s) &= \left(\frac{\lambda}{d(s)(1+C)}\right) \\ d(s) &= \sum_{i=1}^{L_s-1}\left(\sum_{j=1}^{4}\left(L_{i,j}-L_{i+1,j}\right)^2\right)^{\frac{1}{2}} \\ C &= \frac{1}{L_s}\sum_{i=1}^{L_s} C_i \end{aligned} \tag{4}$$

Parameters used in Eq. 4 are identical as in the proposed fitness function. We perform Experiment 1.1 in one of the test plants in Ali's paper and compared the success rates and solution qualities. All the runs are performed in the same GA procedure as stated in the Section 3 with fitness function differed. The inputs of Experiment 1.1 are shown in Table 10.

Fig. 8 shows the result path using the two fitness functions. The elementary operations are displayed in different colors. The red color dotted line denotes the trajectory performed by the target during the swinging operations. The green and blue colored lines stand for the load center trajectory for the luffing and hoisting movements accordingly. The yellow fan represents the rotation of the load. In the path generated with Ali's fitness function, the crane undergoes 67° of swinging, 10° of luffing, 23.74 m of hoisting and 67° of load rotation above the start position. The path consists of 9 configurations. In the path generated with the proposed fitness function, the terrain crane need to conduct 67° of swinging, 0° of luffing, 26.52 m of hoisting and 67° of load rotation. The human operators have 2 less steps to conduct and there is no luffing operation involved. The results show that our proposed fitness function can produce better solution quality.

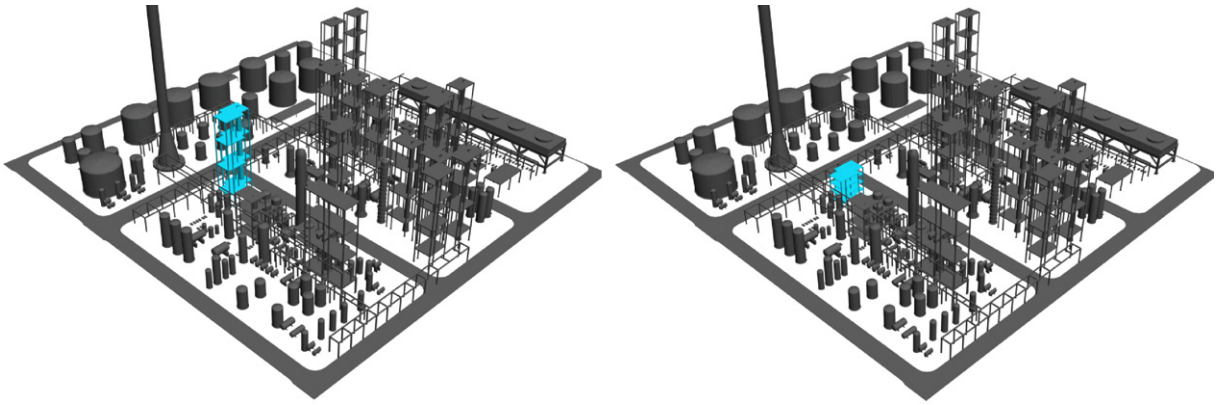
The convergence trends using the two fitness functions are shown in Fig. 9 with the fitness value of the best candidate and the average fitness value of the population during the iterations plotted. The convergence of the GA search with Ali's fitness function shows a highly unstable trend. This instability results in a very low success rate (8%). The convergence with our proposed fitness function shows a much more stable pattern. The search achieves convergence within 250 iterations in this test. The success rate reaches 100% using the proposed fitness function.

Similar comparisons (Experiment 1.2) are conducted in three additional plants as shown in Fig. 10. As indicated in Table 11, the success



(a) Path generated using the C-spaces trategy (b) Path generated using the online strategy (c) Path generated using the hybrid strategy

**Fig. 11.** Trajectory of the load of the result paths using the three strategies in Experiment 2. Green: the C-space strategy; Red: the online strategy; Yellow: the hybrid strategy.



(a) Plant model for Experiment 3.1: Higher blue color obstacle. (b) Plant model for Experiment 3.2: Lower blue color obstacle.

**Fig. 12.** Experiment plant models.

rates using Ali's fitness function of lifting cases in plants 1 and 3 are very low (8% or lower). The success rate of the lifting case in plant 2 using Ali's fitness function is much higher (88%) than in plants 1 and 3. A possible reason is that, the place position of the lifting target in plant 2 is higher than the obstacles, making it easy to achieve collision avoidance. For the lifting case in plant 3 where the obstacles are much higher than the target pick and place positions, it is more challenging to find valid paths. Our proposed fitness function is able to achieve nearly 100% success rate for all the cases.

Table 12 shows the movement units and number of operation steps of the result paths using Ali's fitness function and the proposed fitness function. Although the success rates of the lifting case in plant 2 are both high using the two fitness functions, the solution qualities are quite different. The path generated with the proposed fitness function performs 21 less degrees of luffing while the hoisting height remains similar. In both plant 1 and plant 2, the proposed fitness function can produce paths with 3 less operation steps.

The comparisons indicate that the fitness function use by Ali et al. may not be suitable for the MSPGA framework for single-crane lifting path planning. The proposed fitness function can achieve significant improvement on the success rate of the GA search and in the meantime better solution qualities (fewer motion units and operation steps). The improvements are due to three major reasons in our opinion:

(1) The separation of collision avoidance and optimization of path quality. The fitness value of collision-free paths are much higher than invalid paths. This enables the valid paths fast conquer the population;
(2) Taking into consideration of the human operational conformities and easiness (fewer operation steps involved);
(3) Scaling of the fitness function. Big scaling factors in the fitness function increase the difference between valid paths, which increase the selection pressure in the population and push the GA towards convergence.

### 5.2. Validating the hybrid C-space strategy

To elaborate the benefits of our hybrid C-space strategy (hybrid strategy in short for this section), we apply the proposed algorithm in a complex industrial site to compare the hybrid strategy with the other strategies: C-space and online. The C-space strategy uses the 2D C-space described in [16] and the online strategy uses the analytical swept frontiers to perform online CCD during the GA search as described in Section 4.2. The hybrid strategy uses a 2D C-space to store precomputed collision information for swinging and luffing, and applies the analytical swept frontiers for load movements (which reflect the other two DOFs). Experiment 2 is conducted in a complex plant containing 274,108 vertices and 376,205 triangle faces. With the same set of experiment setting (Table 13), the average computation time, success rates and solution qualities for 100 trial runs are indicated in Table 14. The hybrid strategy spends only half the planning time of the online strategy and obtains similar solution qualities. The preprocessing time is also reduced compared with the C-space strategy. Fig. 11 shows sample result paths generated with the three strategies. While both the online and hybrid strategies manage to rotate the load in order to pass through the plant structure, the C-space strategy needs to pass through the other direction which requires much more movements.

### 5.3. Discussion on parameter design

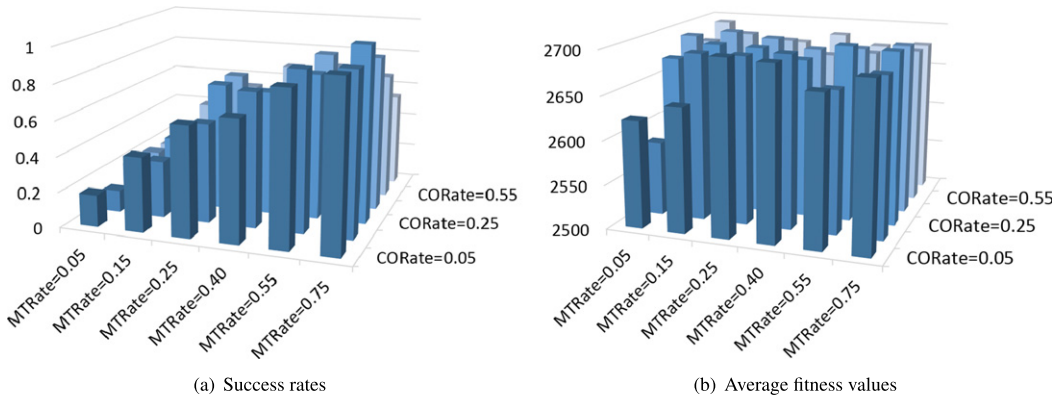
In fact, the selection of parameters, especially the reproduction rates, affect significantly the performance of the algorithm. Here in Experiment 3.1 we conduct a statistical study for the reproductive rates ($r_c$ & $r_m$). Using the plant model as shown in Fig. 12, we take 50 samples of execution for each combination of reproductive rates. We spread the reproduction rates from 0.05 to 0.75. The results, in terms of fitness value of the final solution and the success rates

**Table 15**
Probabilities of finding feasible solutions under different combinations of reproductive rates for Experiment 3.1.

| | $r_m = 0.05$ | $r_m = 0.15$ | $r_m = 0.25$ | $r_m = 0.40$ | $r_m = 0.55$ | $r_m = 0.75$ |
|---|---|---|---|---|---|---|
| $r_c = 0.05$ | 0.18 | 0.42 | 0.62 | 0.68 | 0.86 | 0.94 |
| $r_c = 0.15$ | 0.12 | 0.32 | 0.56 | 0.76 | 0.9 | 0.92 |
| $r_c = 0.25$ | 0.2 | 0.38 | 0.72 | 0.7 | 0.82 | 1 |
| $r_c = 0.40$ | 0.2 | 0.36 | 0.72 | 0.66 | 0.88 | 0.88 |
| $r_c = 0.55$ | 0.18 | 0.46 | 0.6 | 0.74 | 0.76 | 0.72 |
| $r_c = 0.75$ | 0.12 | 0.12 | 0.32 | 0.32 | 0.5 | 0.54 |

**Table 16**
Average fitness value for collision free results under different combinations of reproductive rates for Experiment 3.1.

| | $r_m = 0.05$ | $r_m = 0.15$ | $r_m = 0.25$ | $r_m = 0.40$ | $r_m = 0.55$ | $r_m = 0.75$ |
|---|---|---|---|---|---|---|
| $r_c = 0.05$ | 2621.57 | 2641.26 | 2697.87 | 2695.51 | 2669.43 | 2687.2 |
| $r_c = 0.15$ | 2583.56 | 2689.6 | 2690.29 | 2695.53 | 2661.03 | 2680.24 |
| $r_c = 0.25$ | 2671.82 | 2691.26 | 2690.43 | 2680.13 | 2698.68 | 2695.71 |
| $r_c = 0.40$ | 2690.81 | 2698.15 | 2693.12 | 2683.36 | 2687.22 | 2693.53 |
| $r_c = 0.55$ | 2676.43 | 2688.1 | 2683.17 | 2668.07 | 2673.2 | 2678.88 |
| $r_c = 0.75$ | 2692.9 | 2659.97 | 2673 | 2684.72 | 2673.45 | 2677.55 |




(a) Success rates (b) Average fitness values

**Fig. 13.** Topographic maps of success rates and average fitness values different combinations of reproduction rates in Experiment 3.1.

(probability of obtaining collision free paths), are illustrated in Tables 15–16 and Fig. 13. In this particular plant model, the best solution is achieved with $r_c = 0.25$ and $r_m = 0.55$. Meanwhile, the best ability of collision avoidance is obtained with the crossover rate and mutation rate set as 0.25 and 0.75. We come to a conclusion that, for this particular plant, the best setting of reproduction rates is: $r_c = 0.25$, $r_m = 0.55 \sim 0.75$.

**Table 17**
Probabilities of finding feasible solutions under different combinations of reproductive rates for Experiment 3.2.

| | $r_m = 0.05$ | $r_m = 0.15$ | $r_m = 0.25$ | $r_m = 0.40$ | $r_m = 0.55$ | $r_m = 0.75$ |
|---|---|---|---|---|---|---|
| $r_c = 0.05$ | 0.26 | 0.46 | 0.64 | 0.7 | 0.82 | 0.86 |
| $r_c = 0.15$ | 0.14 | 0.42 | 0.7 | 0.82 | 0.84 | 0.94 |
| $r_c = 0.25$ | 0.26 | 0.64 | 0.58 | 0.84 | 0.96 | 0.96 |
| $r_c = 0.40$ | 0.24 | 0.54 | 0.7 | 0.78 | 0.9 | 0.98 |
| $r_c = 0.55$ | 0.26 | 0.38 | 0.56 | 0.66 | 0.88 | 0.86 |
| $r_c = 0.75$ | 0.08 | 0.44 | 0.54 | 0.42 | 0.46 | 0.6 |

**Table 18**
Average fitness value for collision free results under different combinations of reproductive rates for Experiment 3.2.

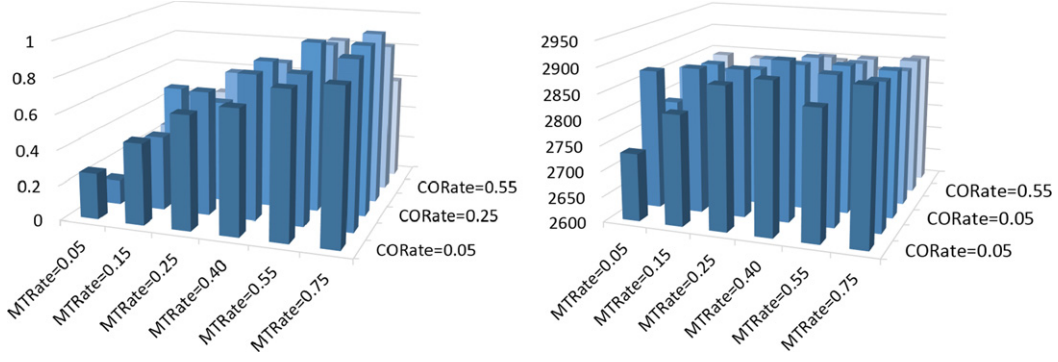
| | $r_m = 0.05$ | $r_m = 0.15$ | $r_m = 0.25$ | $r_m = 0.40$ | $r_m = 0.55$ | $r_m = 0.75$ |
|---|---|---|---|---|---|---|
| $r_c = 0.05$ | 2732.36 | 2816.53 | 2878.26 | 2894.7 | 2852.79 | 2898.74 |
| $r_c = 0.15$ | 2873.98 | 2884.97 | 2889.7 | 2912.74 | 2892.72 | 2887.05 |
| $r_c = 0.25$ | 2792.33 | 2876.66 | 2872.2 | 2887.83 | 2892.21 | 2888.91 |
| $r_c = 0.40$ | 2845.68 | 2844.28 | 2875.95 | 2885.69 | 2879.5 | 2871.27 |
| $r_c = 0.55$ | 2797.59 | 2748.11 | 2808 | 2870.67 | 2871.94 | 2874.16 |
| $r_c = 0.75$ | 2841.87 | 2839.74 | 2834 | 2846.18 | 2832.1 | 2862.73 |

But the result above is not necessarily suitable for other plant environments. We conducted another experiment (Experiment 3.2). We reduce the height of the major obstacles in the previous plant so that the crane does not need to raise its booms to avoid them (Fig. 12). The experiment is conducted with the same set of parameters and the results are listed in Tables 17–18 and Fig. 14. From the data we can see that, the best collision avoidance ability for this occasion is achieved at $r_c = 0.40$ and $r_m = 0.75$. The best solution quality is obtained at $r_c = 0.15$ and $r_m = 0.40$.

In the two experiment cases, we observe similar patterns in the topographic maps. Fig. 13 and Fig. 14 show that larger mutation rates helps to avoid collisions in the MSPGA search. On the other hand, the best solution qualities would be obtained in the center areas according to Figs. 13 and 14. The complexity of the environment between the destinations do affect the shapes of the patterns, especially on the distribution of solution qualities. It may be possible that, the statistical patterns can be represented as functions of the complexity of the environment between the start and end position. Although the performance of the algorithm is somehow dependent on its parameter design, there is a chance to predict the optimum parameter by the pre-analysis of the complexity of the plant environment. This might be a future direction of our investigation.

## 6. Conclusion

Automatic lifting path planning is an important and challenging process in computer-aided lift planning systems. As prior methods are not suitable for fast and effective lifting path planning in complex 3D environments, we propose a new automatic lifting path planning algorithm. We provide a comprehensive mathematical formulation of the lifting




(a) Success rates (b) Average fitness values

**Fig. 14.** Topographic maps of success rates and average fitness values different combinations of reproduction rates in Experiment 3.2.

path planning problem and a customized MSPGA solver for the specified optimization problem. Compared to Ali's fitness function, the proposed fitness function shows great advantage on the success rates. In order to deal with complex environments, a depth map representation of the plant model which is suitable for GPU parallelization is proposed. Based on this representation, an image-space discrete collision detection algorithm and a swept frontier based continuous collision detection algorithm are designed and embedded into the GA search. Finally, to balance the computation time and solution quality, we propose a hybrid C-space strategy for collision checks. This strategy preserves the good solution quality of the online strategy and reduces significantly the preprocessing time of the C-space strategy taking its advantage of runtime efficiency.

The lifting path planning algorithm is able to handle complex plant environments and output safe lifting paths highly optimized in terms of energy cost and human conformity. The result paths are smooth and easier for human operators to conduct with less operation steps.

## Acknowledgment

The authors would like to thank Mr. Lihui Huang, Mr. Yong Chen and many others for their help in this project. Thanks should also go to the supports from PEC Ltd.

## References

[1] OSHA, Statistics & data, https://www.osha.gov/oshstats/index.html 2013 (Online; accessed 19-Dec-2013).
[2] Cranes & Access, A guide to rental rate in 2011, http://www.vertikal.net/uploads/tx_filelinks/ca_2011_9_p1627.pdf Dec. 2011.
[3] Liebherr, Product advantages — mobile crane ltm 1200-5.1, http://www.liebherr.com/AT/enGB/products_at.wfw/id42940/measuremetric/tab3742_1477 2007.
[4] W. Hornaday, C. Haas, J. O'Connor, J. Wen, Computer-aided planning for heavy lifts, J. Constr. Eng. Manag. 119 (3) (1993) 498–515.
[5] K.-L. Lin, C.T. Haas, An interactive planning environment for critical operations, J. Constr. Eng. Manag. 122 (3) (1996) 212–222.
[6] K. Varghese, P. Dharwadkar, J. Wolfhope, J.T. O'Connor, A heavy lift planning system for crane lifts, J. Comput. Aided Civ. Infrastruct. Eng. 12 (1) (1997) 31–42.
[7] S. Chadalavada, K. Varghese, Development of a computer aided critical lift planning system using parametric modeling software, Proceedings of the 2010 International Conference on Engineering, Project, and Production Management (14-C15 October 2010) 2010, pp. 1–12.
[8] J. Olearczyk, M. Al-Hussein, A. Bouferguène, Evolution of the crane selection and on-site utilization process for modular construction multilifts, Autom. Constr. 43 (2014) 59–72.
[9] H. Safouhi, M. Mouattamid, U. Hermann, A. Hendi, An algorithm for the calculation of feasible mobile crane position areas, Autom. Constr. 20 (4) (2011) 360–367.
[10] Z. Lei, H. Taghaddos, U. Hermann, M. Al-Hussein, A methodology for mobile crane lift path checking in heavy industrial projects, Autom. Constr. 31 (2013) 41–53.
[11] Z. Lei, S. Han, A. Bouferguène, H. Taghaddos, U. Hermann, M. Al-Hussein, Algorithm for mobile crane walking path planning in congested industrial plants, J. Constr. Eng. Manag. 141 (2) (2014) 05014016.
[12] L.-C. Lien, M.-Y. Cheng, Particle bee algorithm for tower crane layout with material quantity supply and demand optimization, Autom. Constr. 45 (2014) 25–32.
[13] P. Sivakumar, K. Varghese, N. Babu, Path planning of construction manipulators using genetic algorithms, International Symposium on Automation and Robotics in Construction 1999, pp. 555–560.
[14] M.A.D. Ali, N.R. Babu, K. Varghese, Collision free path planning of cooperative crane manipulators using genetic algorithm, J. Comput. Civ. Eng. 19 (2) (2005) 182–193.
[15] H. Reddy, K. Varghese, Automated path planning for mobile crane lifts, J. Comput. Aided Civ. Infrastruct. Eng. 17 (6) (2002) 439–448.
[16] Y.-C. Chang, W.-H. Hung, S.-C. Kang, A fast path planning method for single and dual crane erections, Autom. Constr. 22 (2012) 468–480.
[17] J. Olearczyk, A. Boufergune, M. Al-Hussein, U.R. Hermann, Automating motion trajectory of crane-lifted loads, Autom. Constr. 45 (2014) 178–186.
[18] S. Kang, E. Miranda, Planning and visualization for automated robotic crane erection processes in construction, Autom. Constr. 15 (4) (2006) 398–414.
[19] S. Redon, Y. Kim, M. Lin, D. Manocha, Fast continuous collision detection for articulated models, Proceedings of the Ninth ACM Symposium on Solid Modeling and Applications, Eurographics Association 2005, pp. 145–156.
[20] Y. Lin, D. Wu, X. Wang, X. Wang, S. Gao, Lift path planning for a nonholonomic crawler crane, Autom. Constr. 44 (2014) 12–24.
[21] NVidia, C programming guide version 3.2, NVIDIA Corporation, Santa Clara, CA.
[22] P. Cai, Y. Cai, I. Chandrasekaran, J. Zheng, Collision detection using axis aligned bounding boxes, in: Y. Cai, S.L. Goei (Eds.), Simulation, Serious Games and Their Applications, Springer Dec 2013, pp. 1–14.
[23] J. Sanders, E. Kandrot, CUDA by Example: An Introduction to General-purpose GPU Programming, Addison-Wesley Professional, 2010.
[24] M. Nowostawski, R. Poli, Parallel genetic algorithm taxonomy, Third International Conference on Knowledge-Based Intelligent Information Engineering Systems., IEEE 1999, pp. 88–92.
[25] M.A. Ismail, Parallel genetic algorithms (pgas): master slave paradigm approach using mpi, E-Tech, IEEE 2004, pp. 83–87.
[26] R. Arora, R. Tulshyan, K. Deb, Parallelization of binary and real-coded genetic algorithms on gpu using cuda, 2010 IEEE Congress on Evolutionary Computation (CEC), IEEE 2010, pp. 1–8.
[27] M. Oiso, T. Yasuda, K. Ohkura, Y. Matumura, Accelerating steady-state genetic algorithms based on cuda architecture, 2011 IEEE Congress on Evolutionary Computation (CEC), IEEE 2011, pp. 687–692.
[28] K. Wang, Z. Shen, A gpu-based parallel genetic algorithm for generating daily activity plans, IEEE Trans. Intell. Transp. Syst. 13 (3) (2012) 1474–1480.
[29] N. Fujimoto, S. Tsutsui, Parallelizing a genetic operator for gpus, 2013 IEEE Congress on Evolutionary Computation (CEC), IEEE 2013, pp. 1271–1277.
[30] P. Cai, Y. Cai, I. Chandrasekaran, J. Zheng, A GPU-enabled parallel genetic algorithm for path planning, in: Y. Cai, Simon See (Eds.), GPU Computing and Applications, Springer 2014, pp. 1–12.