ELSEVIER

Advanced Engineering Informatics 16 (2002) 291–303

ADVANCED ENGINEERING INFORMATICS

www.elsevier.com/locate/aei

Path planning in construction sites: performance evaluation of the Dijkstra, A*, and GA search algorithms

A.R. Soltani*, H. Tawfik, J.Y. Goulermas, T. Fernando

Advanced Virtual Prototyping Group, Centre for Virtual Environment, Information System Institute, University of Salford, Business House, Salford M5 4WT, UK

Received 31 January 2003; revised 20 February 2003; accepted 29 May 2003

Abstract

This paper presents the application of path planning in construction sites according to multiple objectives. It quantitatively evaluates the performance of three optimisation algorithms namely: Dijkstra, A*, and Genetic algorithms that are used to find multi-criteria paths in construction sites based on transportation and safety-related cost. During a construction project, site planners need to select paths for site operatives and vehicles, which are characterised by short distance, low risks and high visibility. These path evaluation criteria are combined using a multi-objective approach. The criteria can be optimised to present site planners with the shortest path, the safest path, the most visible path or a path that reflects a combination of short distance, low risk and high visibility. The accuracy of the path solutions and the time complexities of the optimisation algorithms are compared and critically analysed.

Keywords: Construction site; Path finding; Multi-criteria; Optimisation algorithms

1. Introduction

The construction process requires a set of spaces to execute the construction activities in a safe and efficient manner [1]. Planning construction site spaces to allow for safe and efficient working conditions is a complex and multi-disciplinary task as it involves accounting for a wide range of scenarios. In construction sites, a safe working environment is imperative as it brings about a sustained condition in which site operatives work efficiently [42]. Perry and Hayes [34] and Mustafa and Al-Bahar [32] have identified risks sources central to construction activities to be physical, environmental, design, logistics, financial, legal, political, and operational risks. The most frequent causes of accidental death and injuries are: falls, falling materials and collapses, electrical accidents, and mobile plant [18]. Such risks can be reduced if the use of vehicles and mobile plant are properly managed. In addition, routing of materials affects the cost and the time during construction projects, and major productivity gains in terms of the reduction in wastage and working time can be achieved by planning the site from a logistics perspective [29]. Provision of safe paths could be used to control high-risk situations on the site, for example, where vehicles are reversing or visibility is restricted. Site layouts with site operative routes separate from vehicles, which possess short distance, low risks, and high visibility, can assist site planners to reduce hazards. It is also of great importance to check whether an object in front of an operator can be seen in workplaces or vehicles [21]. Visibility can be considered to be a safety-related measure that allows for better risk perception and avoidance; visibility measures are useful in that maximising visibility can potentially minimise risks.

The issues of organising site vehicles and people's movement within a site have increasingly become a significant area of interest to site planners. Advances in information technology could be used to help planners in their decision-making process by allowing them to express various scenarios that take into consideration several interrelated factors for site layout and path finding. However, site planners who have a key role in the site management currently rely mainly on their experience and intuition when considering the allocations of paths for logistics, vehicles, and site operatives [26,46]. Therefore, there is a need to exploit more computer-based systems to

* Corresponding author.
E-mail address: a.r.soltani@salford.ac.uk (A.R. Soltani).

1474-0346/02/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S1474-0346(03)00018-1

improve risk management in construction sites and in particular, automate routing analysis of people and vehicles during construction.

2. Related work

Recent research into optimum construction site layout organisation [27], site planning [45], site communication [12], and site scheduling [11] has resulted into the development of a number of IT prototypes to aid site-planning tasks. Unfortunately very limited research into path finding on construction site has been conducted. The ability to solve for the optimal path connecting two points in space has been of great importance to engineering [3,24], and more recently to architectural and urban design communities [6]. Traditional methods of determining an optimal path that traverses from one point to another have been based on graph exploration techniques. Solka et al. [41] used Hassoun and Sanghvi's [17] highly parallel unconstrained Dijkstra approach in order to develop a new optimal algorithm in which paths are subjected to turning constraints such that the final solved path contain no turns greater than 45°. In this approach, the same vertices of a grid-like structure graph may be repeated due to constraints causing a path to traverse the same link more than twice, resulting in a significantly long path. In a similar attempt, Boroujerdi and Uhlmann [5] presented an algorithm for computing least cost paths under turn angle constraints. The task was carried out by a suitable transformation of the underlying graph to another graph in which illegal sequences of edges were structurally eliminated. Romeijn and Smith [36] investigated the problem of computing in parallel all pairs of shortest paths in general large-scale directed networks of N nodes using a hierarchical network decomposition algorithms. Aggregating nodes were used to further reduce the number of processors. Duxbury and Dobrin [10] used Dijkstra to find extremal dynamics for a polymer in a random medium. A greedy search was used to capture locally optimal conditions associated with systems that are driven weakly, known as extremal dynamics in physics. In their study, the growth front and minimal path on a square lattice was found using Dijkstra's algorithm. Kei and Srikanthan [22] proposed a routing strategy for real-time multi-casting application that is based on a clustering technique preferred over Dijkstra' algorithms-based routing optimisation techniques in large networks. The advantage of this technique over Dijkstra was that the increase in the size of the networks and degree of cycles resulted in a linear increase in the overall computation time. The time complexity issue associated with Dijkstra's algorithm was also investigated by Refs. [2,31,43,44,48].

For path planning applications using Genetic algorithms (GA), Mercedes Gomez-Albarran et al. [30] proposed a routing strategy that uses Lee's routing algorithm in conjunction with a GA. The algorithm obtains the shortest path between the nodes of a graph while overcoming the GA drawback by obtaining a smaller search space where Lee's algorithm operates. Their work only seeks to optimise a single criterion, which is the travelling distance between nodes. Gen et al. [14] used GA on problems associated with network designs such as a multi-stage process planning, fixed charge transportation, minimum spanning tree, centralised network design, local area network and shortest path. Hurley and Moutinho [20] analysed the potentialities and degree of applicability of GA, tabu search, and simulated annealing (SA) in the field of marketing management. It was reported that site acquisition and retail outlet location is an important consideration in the enhancement of corporate profitability. They used GA for optimum site location analysis. Khoo and Ng [23] used GA to optimise printed circuit board (PCB) component placement by modelling the motion of a component insertion as a three-dimensional asymmetric travelling salesman problem (TSP). The optimisation intended to minimise the total distance travelled by a machine bed during an assembly process. Sandukar and Chen [38] devised a GA-based pipe routing using tessellated objects, abbreviated as GAPRUS. They claimed that the major feature of their GA lies in its ability to incorporate a composite set of variables consisting of real, integer and discrete type variables. The variables generated by the GA were: the pipe lengths, one of the direction cosines of each pipe, the angles of bends between successive pipes and the number of bends along pipes. Qu et al. [35] used GA to optimise the measuring path on a coordinate measuring machine (CMM). The head of a CMM measures a set of sampling point distributed on the curved surface of a workpiece sequentially, and evaluates if these points locate inside a predetermined tolerance limit. The authors reported that this is a time-consuming process with low productivity and proposed to find the shortest points paths between the sampling points when the measuring head of a CMM can be considered as TSP. Seo and Lee [40] addressed the applicability of GA to optimal evaluation of object-oriented queries containing path predicates.

In addition, GA was used for robotic path planning [7,13,25]. It was argued that the number of feasible paths between the initial position and final position of a robot is often very large, and the goal is not necessarily to determine the best solution, but to obtain an acceptable one according to certain requirements and constraints.

The paper is organised as follows: In Section 3 the aims and objectives are introduced. Section 4 will address the design concepts, theoretical frameworks and illustrate working principles associated with Dijkstra, A^* and GA search algorithms. In Section 5 the results from the application of search algorithms to a construction site are presented and quantitatively analysed. Finally, Section 6 concludes the work presented.

3. Construction site path planning approach

The work reported in this paper describes a framework for supporting path planning analysis in construction sites based on multi-criteria evaluation of transportation, safety and visibility measures. This application aims to enable site managers and planners to make strategic decisions regarding the movement of people and vehicles within a site, allowing them to examine possible path scenarios and select the best available route. This study addresses the case of a static site layout in which combinations of the above-mentioned criteria are optimised to present site planners with paths that reflect combinations of short distance, low risk, and high visibility between two site locations. The applicability of three search algorithms, which are Dijkstra, A* and GA, is assessed.

4. Design concepts

The quantitative analysis of a construction site layout requires an analytical framework that provides problem solutions by applying mathematical methods which perform site layout representation, distance formulation, hazard zone modelling, and visibility calculations. The site layout representation transforms a site layout into a multi-dimensional grid of discrete spatial elements containing distance, hazard, and visibility information, as shown in Fig. 1(a).

This multi-layered quantitative spatial structure provides a discrete representation of the path evaluation criteria. The aim is to carry out a multi-criteria evaluation of travelling distance, safety, and visibility in order to find a path between two locations in the site layout such that the resulting path solutions pose the minimum distance, risk and is exposed to maximum visibility (Fig. 1(b)). The weighted sum multi-criteria evaluation is used to create a movement path for a fixed start to goal destination, which can be varied depending upon the weighting allocated to each criterion. The weighting values associated with each criterion demonstrate the manner in which a combination of multi-criteria contributes to an optimum movement path. For example, if the safety aspects need to be signified, a larger weighting value for safety criterion is used relative to other criteria, resulting in a movement path that reflects a greater emphasis on an optimised safety route. The above modelling strategy that leads to an optimal path can be broken into the following key steps: (1) site layout representation; (2) path evaluation criteria; (3) multi-criteria path cost evaluation; (4) path search optimisation. These steps are outlined below.

4.1. Site layout representation

In order to present a site layout for analysis, the site layout is subdivided into cells on a two-dimensional grid. A grid node (P) is located at a cell's centre-point is allocated to each cell as shown in Fig. 2(a). This method of space mapping generates a set of discrete nodes that cover the entire construction site domain, as shown in Fig. 2(b) in which dark areas indicate the presence of site objects. The spatial criteria: distance, safety, and visibility are presented by a set of numerical values attributed to each node. This discrete representation of the site layout can be used to develop a directed graph whose vertices and edges are the location of the discrete nodes and the associated numerical node values, respectively (Fig. 2(c)). The edges of the directed graph contain the numerical values of distance, safety and visibility associated with each vertex.

The distributions of the path evaluation criteria across the site layout are obtained by evaluating the numerical values of each criterion at the grid nodes, using distance, hazard, and visibility models.

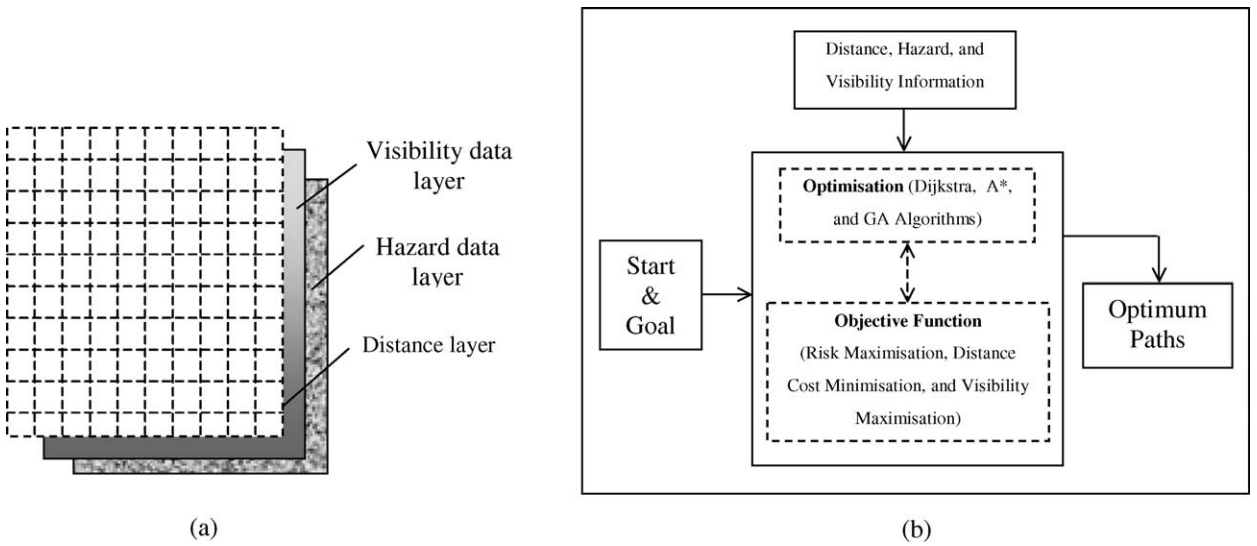


Fig. 1. The design conceptualisation using (a) multi-layered grid structure, (b) structure of path finding application.

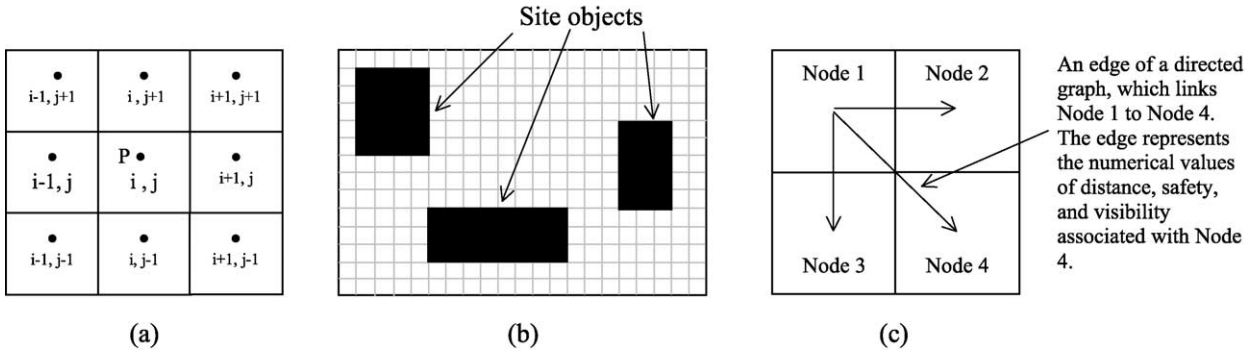


Fig. 2. (a) Grid point representation, (b) a sample of a discretised site layout, (c) a directed graph developed by the discrete representation of the space layout with black shapes highlighting site objects.

4.2. Path evaluation criteria

The grid nodes of the discretised site layout hold numerical values corresponding to distance, safety, and visibility. These are spatially related criteria since they convey information regarding space. The formulations of these criteria are presented below.

4.2.1. Distance evaluation

This is the Euclidean distance between the grid nodes. In the case of Dijkstra and A*, the distance between two adjacent grid nodes (Fig. 2(c)) are 1 for horizontally and vertically aligned grid nodes, and $\sqrt{2}$ for diagonally aligned grid nodes. In the case of the GA, the distance is calculated between the endpoints of every segment of the path, as described in Section 4.4.3.

4.2.2. Hazard zone modelling

The hazard zone modelling part determines the hazard zones associated with the presence of site objects such as cranes, vehicles and equipment, according to their variable degree of risk and dimension. Hazard zones in this application are represented by a number of layers that vary in terms of the size of the hazard area surrounding the object and the distribution of the degree of hazard across the hazard layers. The extent and intensity of hazard from an object's boundary is defined by the hazard length (L) and the hazard level (H), respectively, as shown in Fig. 3. This allows heuristic generation of hazard values for different site objects on a construction site, assuming hazard is present across all sides of a site object by a similar degree. The hazard values are normalised between [0,1].

The distribution of the hazard level across the hazard area represented by three main heuristics:

- constant hazard distribution where the hazard area has a constant hazard level H_x (Fig. 3(a)), a typical example may occur when a ditch is excavated and the entire excavated area exposes the same hazard level;
- linearly decreasing hazard distribution in which the hazard level (H_x) have maximum values at the object's boundary and decrease linearly according to the equation

$$H_x = \frac{H_{\max} - (H_{\max} - H_{\min})x}{L}$$

where x is the distance from the object's boundary and $H_{\max}$ and $H_{\min}$ are maximum and minimum hazard levels, respectively (Fig. 3(b)), $0 < H_{\min} < H_{\max}$. A mobile plant such as a tower crane can produce a varying degree of risk being maximum at the tower while decreasing linearly with movement along the boom length;

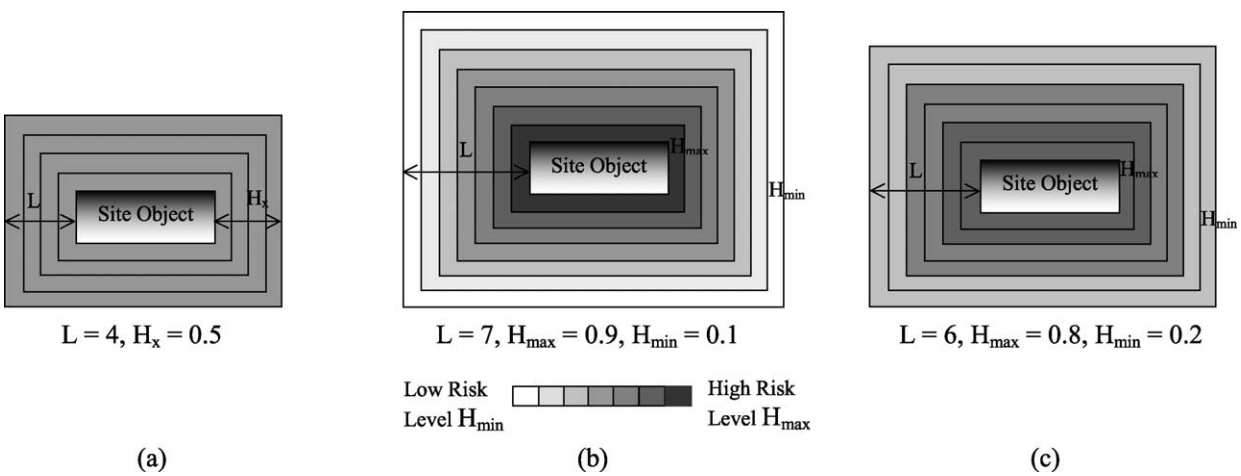


Fig. 3. Hazard zone with (a) constant hazard distribution, (b) linear hazard distribution, (c) non-linear hazard distribution.

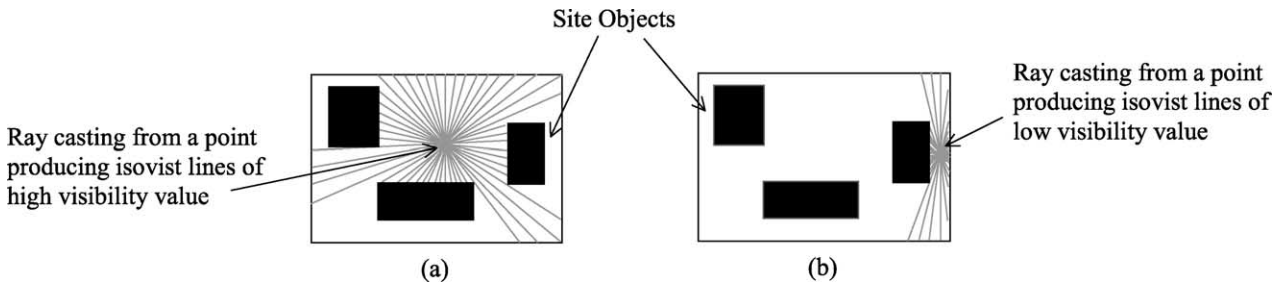


Fig. 4. Visibility analyses of spatial layouts using isovists lines for (a) a high visibility point, (b) a low visibility point.

- non-linearly decreasing hazard distribution in which the hazard level (H_x) has maximum values at the object boundary and decrease non-linearly according to the equation (Fig. 3(c)).

$$H_x = \frac{H_{\max} - (H_{\max} - H_{\min})x^2}{L}$$

A non-linear hazard distribution may arise when falling from heights occurs in which the degree of injuries non-linearly intensifies as the height increases.

4.2.3. Visibility calculation

This criterion determines the visibility value from a fixed position in the site. The visibility model consists of a set of lines of sight, by which objects in the construction sites can be seen. The lines of sights which are referred to as isovists lines [4], can be defined as the geometry obtained by casting light rays in all directions from a fixed position in space. The isovist lines are obtained when the site objects or the site boundaries intersect the rays, as shown in Fig. 4. The isovist lines are summed up and divided by the number of rays in order to calculate the average visibility values. By applying the isovist analysis on a proposed site layout, regions in the site that offer different levels of visibility are identified. Fig. 4 shows the results of applying visibility analysis using isovists to a single point on two different locations within a site layout. Fig. 4(a) shows a high visibility point with longer isovists lines compared to a low visibility point shown in Fig. 4(b).

In the context of a construction site, spaces with higher levels of visibility can be used by the site planner to make more informed decisions regarding high visibility areas and relatively isolated ones.

4.3. Multi-criteria path cost evaluation

A composite objective value of a grid node P_k on the site layout, is obtained by summing the normalised values of distance, safety and visibility through the following weighted-sum method

$$C(P_k, P_{k+1}) = \lambda_d C_d(P_k, P_{k+1}) + \lambda_s C_s(P_k) + \lambda_v C_v(P_k)$$

where P_k and P_{k+1} are a pair of adjacent nodes and λ_d, λ_s and λ_v are the distance, safety and visibility weights at each grid node, respectively. $C_d(P_k, P_{k+1})$ is the binary distance function and $C_s(P_k)$ and $C_v(P_k)$ are the unary functions of safety and visibility. In the case of Dijkstra and A*, the path cost evaluation function $F(\mathbf{x})$ is given by

$$F(\mathbf{x}) = \sum_{k=1}^{m-1} C(P_k, P_{k+1})$$

where $\mathbf{x}$ is the entire path, m is the number of grid nodes residing between the start and the goal nodes.

The path cost evaluation function $F(\mathbf{x})$ is the summation of the composite objective values of the grid nodes associated with the grey cells, as shown in Fig. 5. The grid nodes that are occupied by a site object, shown as black cells in Fig. 5, are assigned very large $C(P_k, P_{k+1})$ values in order to incur a high penalty on the evaluation function.

For example, the distance cost function for the path $\mathbf{x}$, in Fig. 5, is the length of the path between the start and the goal node, which is 9.65 m for non-normalised distance values. Similarly, if the normalised safety values were assumed to be 0.5 at every grid node, then the safety cost function for the path $\mathbf{x}$ would be 4.5.

In the case of the GA, the path cost evaluation function $F(\mathbf{x})$ is slightly different to that of Dijkstra and A* and is described in Section 4.4.3.

4.4. Path search optimisation

In general, there are two types of optimisation algorithms: (a) algorithms that are deterministic whose behaviour can be completely predicted from the input and have specific rules for moving from one solution to the other; (b) algorithms that are stochastic in nature, with probabilistic search rules.

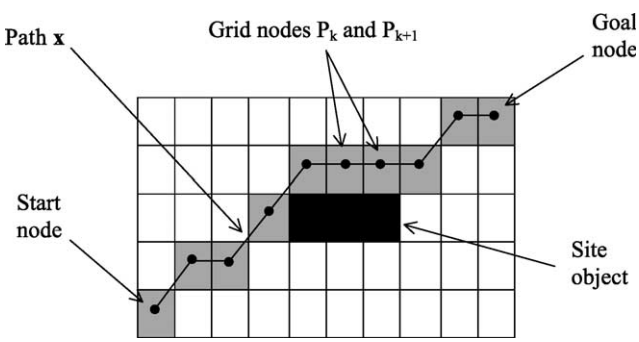


Fig. 5. Path cost evaluation function $F(\mathbf{x})$ for Dijkstra and A*, the grey cells contain the grid nodes that their values provide minimum safety and maximum visibility measures.

For the purpose of this application, there is a need for optimisation techniques that minimise the risk level along the movement paths and at the same time provide a balance between distance (transportation cost) and visibility requirements. The optimisation, in this study, is carried out by three optimisation techniques: two deterministic search methods known as Dijkstra and A*, and GA randomised search. The algorithms' performances are quantitatively compared.

4.4.1. Dijkstra

Dijkstra [9] is an optimisation algorithm that is mainly used for determining the shortest paths. Dijkstra algorithm is an uninformed search algorithm for finding shortest paths that rely purely on local path cost and provides a shortest path from a starting node to a goal node in a graph. Dijkstra algorithm modifies the breadth-first strategy by always expanding the lowest-cost leaf node first as measured by the path cost $g(n)$, rather than the lowest-depth node, where $g(n)$ gives the path cost from a start node to a goal node n. Dijkstra algorithm is complete and its time complexity is the same as for breath-first search [37].

4.4.2. A*

The A* star algorithm was developed by Hart et al. [16]. The algorithm uses a heuristic function $h(n)$ to estimate the cost of the lowest cost path from a start node to a goal node plus the path cost $g(n)$, and therefore the search cost $f(n) = g(n) + h(n)$. Using f, g, and h values, the A* algorithm will be directed towards the goal and will find it in the shortest possible route. The restriction is to choose an admissible h function that does not overestimate the cost to reach the goal [37]. A* is only admissible if the heuristic function h always gives the exact distance to the goal. In practice, this may be impossible and if the heuristic function overestimates the real distance to the goal by more than a certain value, then the algorithm will find a solution that costs more than the optimal solution. Therefore, in terms of time complexity, completeness and optimality the quality of the heuristic function plays an important role. In this study two heuristic functions were used:

- the straight-line distance heuristic indicating the first node to be expanded is the one closer to the goal node and is mainly used for distance cost search. The straight-line distance between every node and the goal node can be easily obtained by using their Euclidean distances;
- the cumulative line-nodes heuristic sums up the safety and visibility cost values of grid nodes that are laid on a straight line between every grid node and the goal node and is used for safety and visibility measures. This requires an iterative operation that looks at all the grid nodes and selects the ones that are laid on a straight line between a single grid node and the goal node, and then sums their safety/visibility values. This process is repeated for every node relative to the goal node.

4.4.3. Genetic algorithms

Genetic algorithms (GA) are powerful random search methods that adopt mechanisms of natural selection and genetics in their search for optimal solutions [15]. GAs were originally developed by Holland [19], and seek optimum solutions from a population of points in a search space in order to find global solutions. A GA starts with a population of chromosomes representing possible problem solutions as coded strings such as binary, integer, float strings and an objective function to evaluate the performance of a problem solution according to problem-specific optimisation criteria. GA generates successive and improved populations of solutions over a number of generations by applying the main genetic operations of reproduction, crossover, and mutation. Reproduction allows chromosomes of higher performance to contribute with higher probability to the population of the next generation. This is done by a biased random selection, i.e. strings with a probability proportional to their fitness are selected. Then, the selected strings are mated randomly and each pair exchanges some substrings randomly according to a crossover site also chosen randomly to produce two off-strings. The concept of crossover is that the genetic material of different members can be combined to produce an individual that would benefit from the strengths of both parents. Finally, mutation operator flips an arbitrary gene in the chromosomes with a given probability to maintain diversity in the population. These three steps are repeated to create a new generation and continue until some stopping condition is reached, such as a maximum number of generations, an acceptable approximate solution, or any application-specific criterion. In the context of this study, GA searches for a number of intermediate path points that create a sequence of segments assembling an optimum path (Section 4.4.3.2) between the start and the goal nodes in the site layout.

4.4.3.1. Chromosome representation. The GA chromosome was selected to be an integer string that encodes the two-dimensional geometrical coordinates of a number of grid nodes of the discrete site layout, which correspond to intermediate points between the start and the goal locations, defining a segment sequence that forms the path. For a path consisting of three intermediate points, the corresponding GA chromosomes representation can be shown in Fig. 6. S and G are the start and goal nodes, respectively. The location co-ordinates of the intermediate path points are given by x and y, which form chromosome genes denoted by g.

4.4.3.2. Objective function. The objective function plays a key role in the reproduction process in terms of obtaining feasible and high fitness solutions. It consists of two parts, the objective and the penalty. The first part deals with minimising the overall path cost between the successive intermediate points and, consequently, with finding the shortest path between the start and the goal nodes without considering constraints. The second part takes into account

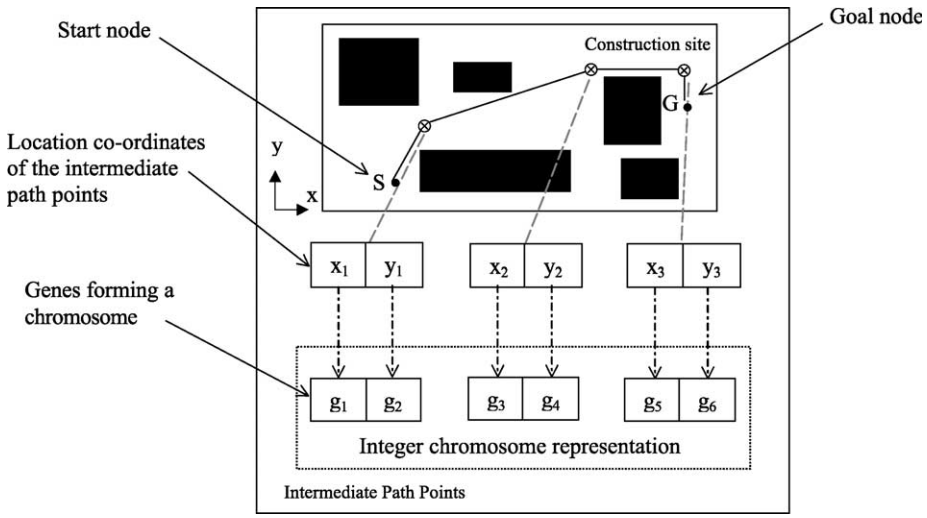


Fig. 6. GA solution representation of a site path.

the constraints in the case where the path passes through site objects. This penalty function imposes a large value proportional to the number of grid cells that lie on the site object and coincide with the path **x**.

For the case of objective part, the path cost evaluation function is first calculated for every individual segment via the summation of safety and visibility values of the grid nodes associated with the grid cells crossed by the segment, as shown in Fig. 7. Then, the overall evaluation function $F(\mathbf{x})$, is computed by the summation of all the segments forming the path. This can be formulated as

$$F(\mathbf{x}) = \sum_{l=1}^{q} \sum_{k=1}^{m_l - 1} \lambda_s C_s(P_k) + \lambda_v C_v(P_k)$$

where q is the number of path segments and m_l is the number of grid cells crossed by each individual path segment l. In the case of the distance measure, the Euclidean distance of every individual path segment is calculated and the overall path cost function is the summation of straight-line segments forming the path.

4.4.4. Time complexities of the algorithms

The computational complexity of the above algorithms is an aspect that should be considered in practical applications where a prerequisite for a successful algorithm is the capability to deliver high performance solutions in a minimal time. A useful measure of algorithm complexity is the execution time, which is often estimated or predicted. An efficient algorithm is the one whose worst-case running time, is bounded by a polynomial of the problem size. To make the time measure independent of a specific computer, time could be measured using a quantity called steps [39]. The total number of steps taken by an algorithm can be characterised as a function of the input size, known as complexity function $T(n)$, where n is the input size. The complexity function $T(n)$ is a member of a set called complexity category $\theta(n)$, so that $T(n) \in \theta(n)$. Both Dijkstra and A* are quadratic-time algorithms $\theta(n^2)$ [33]

$$T(n) = 2(n-1)^2 \in \theta(n^2)$$

where n is the number of vertices of the graph data structure shown in Fig. 2(b), i.e. number of the grid points.

GA performance is usually measured by the number of fitness function evaluations carried out during the course of a GA run. For fixed population sizes, the number of fitness function evaluations is given by the product of population size by the number of generations [28]. Assuming the crossover probability is one, the GA's time complexity is measured as a function of the problem size, population size,

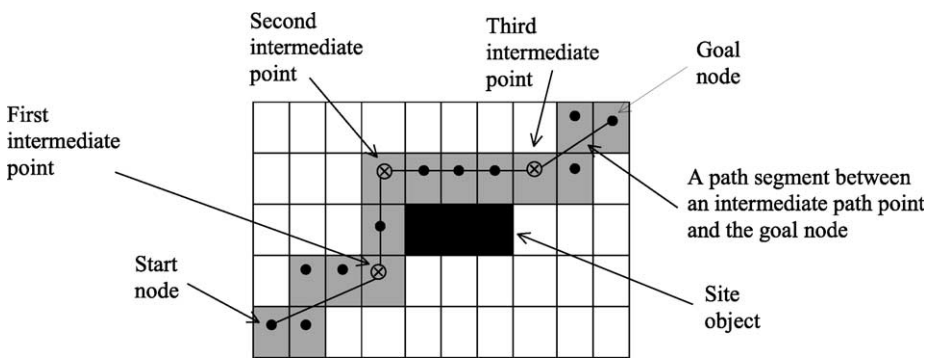


Fig. 7. Evaluation of the GA's objective function for a path consisting of three intermediate path points (four path segments) for safety and visibility measures.

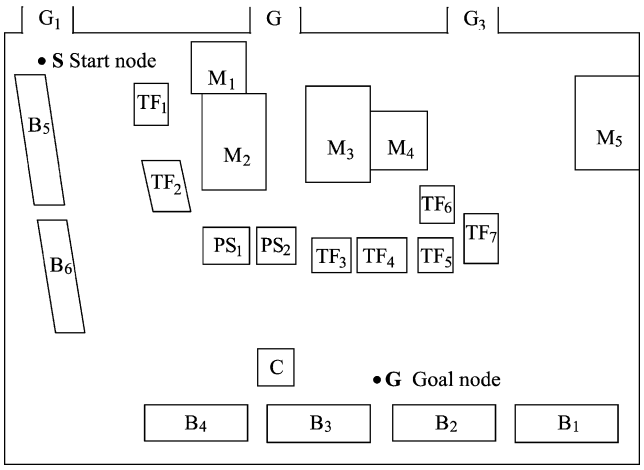


Fig. 8. The site layout of the Skanska construction site under investigation.

number of generations, where the problem size in this study is the number of intermediate path points. Therefore, the time complexity function T is given by

$$T = O(\text{No. of intermediate points} \times \text{Population size} \times \text{No. of generations})$$

where O symbolises the order of function.

The efficiency of GA algorithm can also be analysed by estimating the number of grid points visited if an exhaustive search had been carried out, i.e. the size of the search space. For an n by m grid structure that can be represented by a directed graph with $N = n \times m$ vertices, the combinatory of the search space can be estimated by the permutation of N grid points taking k vertices as:

$$NP_k = k!\binom{N}{k} = \frac{N!}{(N-k)!}$$

5. Experiment and results

An initial site layout showing the site boundaries and the site objects, was generated. It was assumed that the site boundary, temporary facilities and other site objects can be represented by rectangular shapes. A site typically contains spaces for the following objects: Buildings (B), Cranes (C), Temporary Facilities (TF) such as management and staff offices, workers facilities and workshops, other Objects (O).

In order to evaluate the performance of the three search algorithms, a two-dimensional approximation of the real Skanska construction site layout was considered. The Skanska construction project was concerned with the development of residential apartments in Aalborg, Denmark. The site layout contains seven temporary facilities (TF_1, TF_2, TF_3, TF_4, TF_5, TF_6, and TF_7), six buildings (B_1, B_2, B_3, B_4, B_5, and B_6), two parking spaces (PS_1 and PS_2), three site gates (G_1, G_2, and G_3), five materials spaces (M_1, M_2, M_3, M_4, and M_5), and one crane (C) as shown in Fig. 8. Consider a material-delivery scenario where building materials are carried by lorries from a start point S, near the site gate G_1, to a goal point G in the vicinity of buildings B_1, B_2, B_3, and B_4 (Fig. 8). The issues involved with this scenario can be classified as: (a) materials need to be transported as quickly as possible to the required destination; (b) lorries need to use open and visible passages; (c) site operatives need to avoid areas associated with high degrees of risks; (d) a contingency lorries exit route via the site gate G_2 avoiding site congestion.

The hazard model (Section 4.2.2) defines a hazardous area that was non-linearly distributed around the crane object, as shown in Fig. 9(a). In addition, the visibility distribution was obtained by applying isovist analysis (Section 4.2.3) on the site space layout and identifying the various visibility regions, as shown in Fig. 9(b).

Two measures are typically used to compare algorithms quantitatively: (1) the time complexity of the algorithms, and (2) the quality of solution [47]. The comparison is carried out by examining the paths using the following criteria: (1) a distance criterion; (2) a safety criterion; (3) a visibility criterion, and (4) a combination of distance, safety, and visibility measures with 0.25, 0.5, and 0.25 weighting, respectively. The results emerging from the weighted sum multi-criteria path evaluation for the three algorithms are

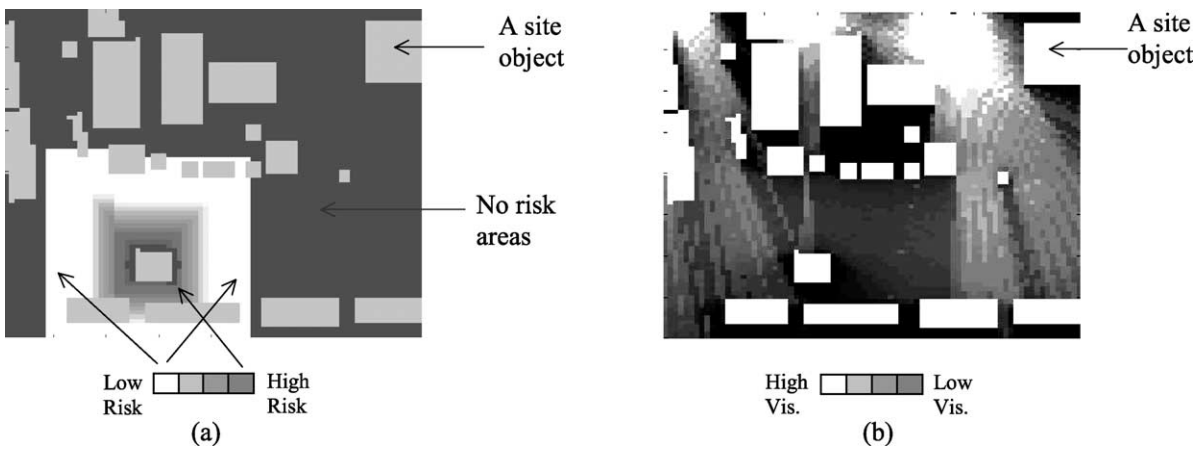
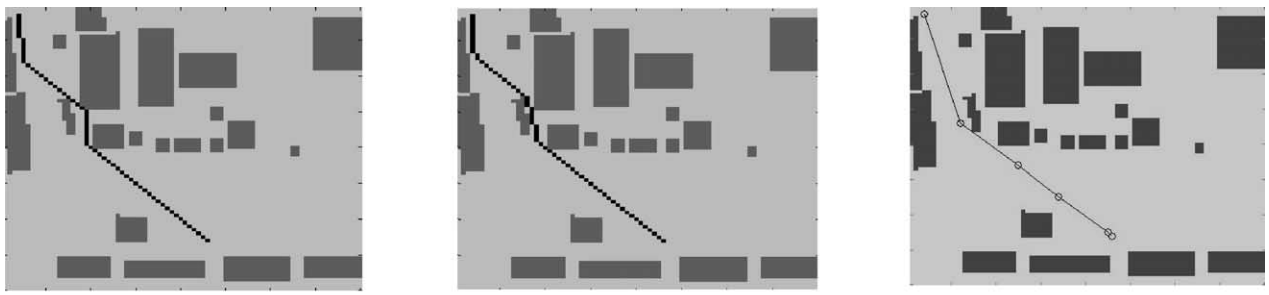


Fig. 9. Graphical representations of (a) safety, and (b) visibility distribution.



(a) (b) (c)

Fig. 10. Optimum paths resulting from the distance criterion.

Table 1
Comparison of the algorithms path cost solution

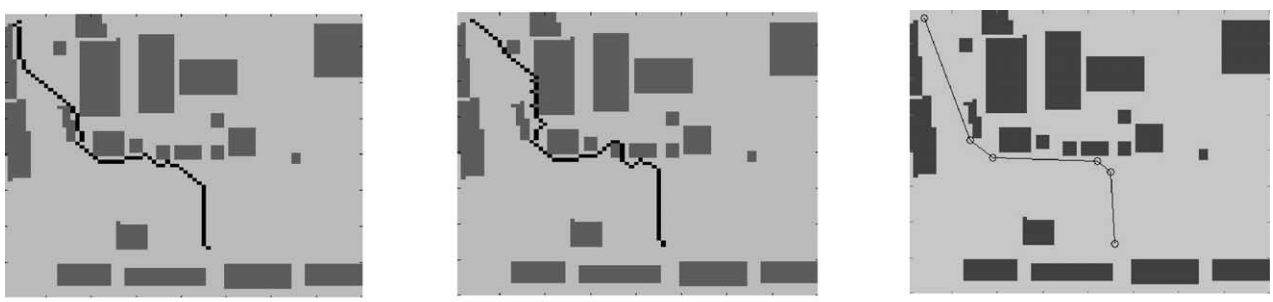
Path evaluation criteria	Dijkstra path cost solution	A* path cost solution	GA path cost solution
Distance minimisation	80.4	80.4	78.6
Safety minimisation	41.5	41.6	43.3
Visibility maximisation	43.5	43.7	42.5
Combined optimisation via G_1	35.3	43.2	39.5
Combined optimisation via G_2	29.7	38.9	30.2

presented, in which the Dijkstra, A* and GA results are designated by (a), (b) and (c), respectively. The algorithms performances relative to each other in terms of solution accuracy, search strategy, and execution time are examined.

The evaluation of the shortest path reveals that both Dijkstra and A* produced very similar paths, Fig. 10(a) and (b), having an identical cost as shown in Table 1. GA path evaluation with four intermediate path points, Fig. 10(c), results in a shorter cost due to the fact that GA's objective function measures the summation of straight-line segments between four intermediate path points (Section 4.4.3). The safest paths shown in Fig. 11(a)–(c), demonstrated that all algorithms avoid the areas associated with the high risk around the crane (Fig. 9(a)). Table 1, shows the error difference in cost solution is significantly negligible between Dijkstra and A*, and only accounts for 4% between Dijkstra and GA, for the case of safety minimisation.

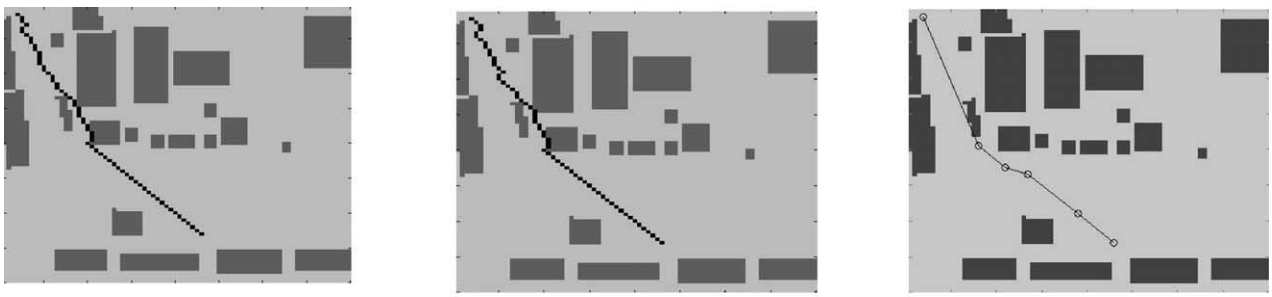
In addition, in Fig. 11, the three methods produced different paths. The optimal path is the one created by Dijkstra algorithm. The A* path is nearly optimal but behaves in an unrealistic manner. The GA's path, although is not optimal, but produces a smooth and less detailed path in a shorter time compared to that of Dijkstra and A*. The GA uses significantly lower number of intermediate points (Table 3) cutting down on the path details and at the same time reduces the time that takes to create the path. Therefore, it is a balance between the solution accuracy and the time that account for the choice of different algorithms, in particular, for large-scale problems. The path finding application also aims to ultimately serve as part of a virtual simulation tool for site planners to experiment with a variety of path scenarios in a near real-time manner, therefore, A* and GA can be used to carry out higher search speed finding near-optimum solutions.

The most visible paths are shown in Fig. 12, however, both Dijkstra and A*, Fig. 12(a) and (b), captured the narrow and less visible gap (Fig. 9(b)) that exists between the temporary zone (TF_2) and the material storage (M_3); both use a greedy search strategy. Greedy strategies do not always yield optimal results in general, but Dijkstra's algorithm always computes the lowest cost paths [8]. Although Dijkstra's path goes through the narrow zone in



(a) (b) (c)

Fig. 11. Optimum paths resulting from the safety criterion.



(a) (b) (c)

Fig. 12. Optimum paths resulting from the visibility criterion.

Fig. 9(b), its overall path cost solution is guaranteed to be minimum.

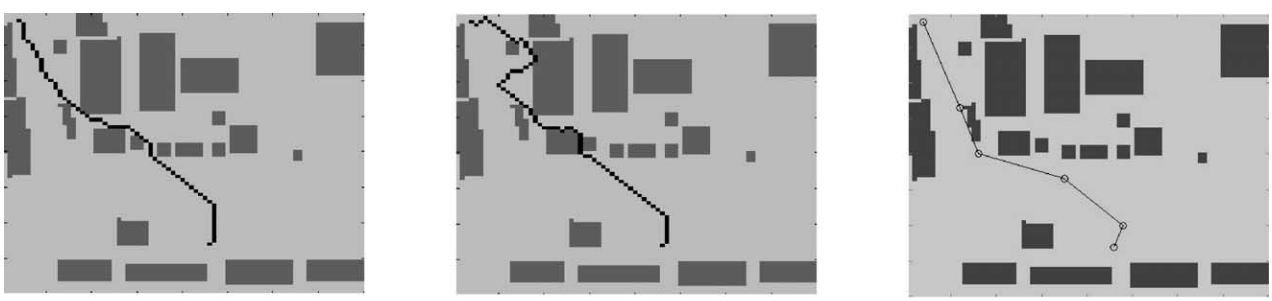
GA also produces an accurate solution (Table 1) avoiding the narrow zone, as shown in Fig. 12(c). The GA's path avoids the narrow gap that is located between TF_2 and M_3, indicating that the GA was not trapped into a local optimisation. In addition, the GA's path has slightly lower cost than both Dijkstra and A^* resulting from the formulation of the GA's path cost function which imposes slight discrepancies on the evaluation of the GA's path cost solution compared to that of Dijkstra and A^* (Section 4.3.3). For multi-criteria path analysis, Dijkstra's creates the lowest cost path, as shown in Fig. 13(a). GA, Fig. 13(c), produces considerably more accurate solutions than A^*, as shown in Table 1. The heuristic function associated with A^* tends to direct search towards low visibility areas (Fig. 13(b)) resulting in less accurate solution (Section 4.4.2). In addition, A^* in Fig. 13(b), produces path that pass through low visibility areas by getting trapped into local optimisation as a result of adopting greedy search strategy. Fig. 14 shows the paths that could also be used by the lorries to enter or exit via the site gate G_2, dealing with the congestion on the site. The Dijkstra's path in Fig. 14(a) uses the gap between PS_2 and TF_4 to get to the gate clearly avoiding the path shown in Fig. 13(a). The A^*'s path, Fig. 14(b) also avoids the path associated with the site gate G_1, but significantly increases the travelling distance. The GA's path tries to avoid the path in Fig. 13(c), but at some point the paths intersect each other. The path cost solutions for Fig. 14 indicate that both Dijkstra and GA have produced low cost paths.

In terms of time complexities (Section 4.4.4), both Dijkstra and A^* are quadratic-time algorithms and the time complexity function $T(n)$ is

$$T(n) = 2(n-1)^2$$

where n is the number of vertices of the graph data structure shown in Fig. 2(c), i.e. number of the grid points. If the execution time of processing one computational step is 10^{-6} s (Table 2), it can be noted that as the site dimension (grid size) increases, the time complexity raises to the extent that the search requirements become computationally prohibitive.

In addition, the execution time associated with A^* can be significantly influenced by the type of heuristic functions. (Sections 4.4.1 and 4.4.2). In terms of search strategy, Dijkstra algorithm is an uninformed search algorithm for finding shortest paths that relies purely on the local path cost. While A^* is an informed search and uses a heuristic function to estimate the cost of the lowest cost path. Both Dijkstra and A^* select the next node to expand without taking into consideration whether this will be best in the long run. However, in terms of time complexity, the nature of the A^* heuristic function plays an important role. In this application, the straight-line distance heuristic function (Section 4.4.2) decreases the time complexity by focusing the search, and therefore, reducing the search space whereas the cumulative line-nodes heuristic function significantly increases the execution time, as shown in Table 4,



(a) (b) (c)

Fig. 13. Optimum paths resulting from a combination of safety, distance, and visibility criteria via gate G_1.

Fig. 14. Optimum paths resulting from a combination of safety, distance, and visibility criteria via gate G_2.

outweighing the benefits that could otherwise have been achieved by the comparatively A*'s quicker search ability.

The GA's time complexity function (Section 4.4.4) T is given by

$$T = O(\text{No. of intermediate points} \times \text{Population size} \times \text{No. of generations})$$

Table 3 tabulates the time complexities for a range of intermediate path points using 100 populations and 200 generations. The number of intermediate points can vary from 1 to 9 depending on the dimension and complexity of the spatial site layout, the straight distance between the start and the finish locations, and the number of obstacles that exist between the start and the finish locations. It can be noted from Table 3 that the predicted GA's execution time is considerably less compared to that of both Dijkstra and A*. In addition, the heuristic penalty function associated with the GA could adversely affect the execution time. However, this adversity does not exponentially grow with the grid size and the overall GA performance remains intact. The efficiency of the GA algorithm can also be analysed by estimating the number of grid points visited if an exhaustive search had been carried out (Section 4.4.4). The factorial growth associated with the exhaustive search, produces 4.1×10^7 solutions that need to be visited by an exhaustive search for an 80 by 80 grid dimension having 6400 grid nodes. The GA with nine intermediated points having 100 population size and 200 generation with crossover probability of 1, generates 360,000 potential solutions that need to be operated on to finally select the best ones. With a lower number of intermediate path points and the crossover probability less than one, the search space is even further reduced. Therefore, the search speed can significantly benefit from the GA application.

Table 2
Predicted execution times with the varying site grid size for Dijkstra and A*

Grid size (square)	Number of vertices (n)	Time complexity ($T(n) = n^2$)	Predicted execution time
20	400	1.6×10^5	0.16 s
40	1600	2.56×10^6	2.56 s
80	6400	4.096×10^7	40.96 s
160	25,600	6.5536×10^8	655.36 ≈ 11 min
320	102,400	1.048×10^{10}	10,480 ≈ 175 min

Table 3
Predicted execution times with the varying number of intermediate points for GA

Number of intermediate path points	Chromosomes size	Time complexity (T)	Predicted execution time (s)
3	6	120,000	0.12
6	12	240,000	0.24
9	18	360,000	0.36

The actual execution times for Dijkstra, A* and GA in the case of 40 by 40 and 80 by 80 grid size are given in Table 4. It can be seen that Dijkstra and A* with straight-line distance heuristic perform the same in terms of actual execution time for relatively small number of vertices. The A* with cumulative line-nodes heuristic raises the actual execution time by nearly three folds. An INDIGO2 SGI machine of 175 MHz speed with IP28 processor was used to compute the actual execution time, which is comparable to a Pentium 200 PC.

The paths shown in this paper were used to evaluate the performance of algorithms particularly by focusing on the detailed path configurations not paying a great deal of attention to how feasible the predicted paths are. The Skanska construction site case study gave us this opportunity to carry out this task. The actual paths of lorries tend to avoid the narrow gaps on the site and the predicted paths could capture that if certain constraint had been applied to the algorithm's cost function. The extension to this work will aim to consider this constraint.

Table 4
Actual execution times for two grid size for Dijkstra, A* and GA

Grid size	Actual execution time (s)			
	Dijkstra	A* with straight-line heuristic	A* with cumulative line-nodes	GA
40	7	7	22	5
80	120	120	456	51

6. Conclusions

This study illustrated the potential of deterministic and probabilistic search algorithms in addressing the site path planning issues with multiple objectives. The application can generate the shortest path, low risk path, most visible path, and finally the path that reflects a combination of low risks, short distance, and high visibility between two site locations. Dijkstra algorithm can find optimal solutions to problems by systematically generating path nodes and testing them against a goal, but becoming inefficient for large-scale problems. A* can find optimal and near to optimal solutions more efficiently by directing search towards the goal by means of heuristic functions, reducing the time complexity substantially. Dijkstra and A* algorithms carry out a greedy search method arriving at a solution by making a sequence of choices, each of which looks for the best at the time without considering the potential drawbacks of making such a choice. These algorithms suffer from the curse of dimensionality effect, which limits the Dijkstra's and A*'s operation to small and medium problems. However, A* could produce solutions that are locally optimal.

Probabilistic optimisation approach based on GA generates a set of feasible, optimal, and close-to-optimal solutions that captures globally optimal solutions. GA operators exploit the similarities in string structures to make an effective search. Therefore, good regions of the search space get exponentially more copies and get combined with each other by the action of GA operators and finally form the optimum or a near-optimum solution in substantially less time. The GA's performance limitations are mainly related to obtaining less accurate solutions and the time-consuming fine-tuning process to guide the search. The future avenues for this work includes investigation of the applicability of fuzzy based multi-criteria evaluation, and hybrid optimisation search algorithms.

Acknowledgements

This work is part of the EU funded DIVERCITY project (IST-1999-13365). The authors would also like to acknowledge the contribution of Jens Ove Skjaerbaek from COWI (Denmark) for providing the Skanska construction site case study.

References

[1] Akinci B, Fischer M, Levitt R, Carlson B. Formalisation and automation of time-space conflict analysis. J Comput Civil Engng 2002;6(2):124–35.
[2] Akkanen J, Nurminen JK. Case study of the evolution of routing algorithms in a network of planning tool. J Syst Software 2001;58: 181–98.
[3] Barraquand J, Langlois B, Latombe J. Numerical potential field techniques for robot path planning. IEEE Trans Syst Man Cybernet 1992;22(2):224–41.
[4] Benedikt ML. To take hold of space: isovists and isovists fields. Environ Plann 1979;B6:47–65.
[5] Boroujerdi A, Uhlmann J. An efficient algorithm for computing least cost paths with turn constraints. Inform Process Lett 1998;67:317–21.
[6] Charles SH, Kincho HL, Latombe J, Kunz J. A performance-based approach to wheelchair accessible route analysis. Adv Engng Inform 2002;36:53–75.
[7] Chen M, Zalzala A. A genetic approach to motion planning of redundant mobile manipulator systems considering safety and configuration. J Robotic Syst 1997;14(7):529–44.
[8] Cormen TH, Leiserson CE, Rivest RL. Introduction to algorithms. Massachusetts: McGraw-Hill; 1990.
[9] Dijkstra EW. A note on two problems in connection with graphs. Numeriche Mathematik 1959;1:269–71.
[10] Duxbury PM, Dorbin R. Greedy algorithms in disordered systems. Physica A 1999;270:263–9.
[11] Elbeltagi E, Hegazy T, Hosny AD, Eldosouky A. Schedule-dependent evolution of site layout planning. Constr Manage Econ 2001;19: 689–97.
[12] Fernando T, Kahkonen K, Leinonen J, Murray N, Tawfik H. Facilitation of collaborative communication for building construction with virtual reality technology. Conference on Applied Virtual Reality in Engineering and Construction Application, AVR II & CONVR, Gothenberg 2001; p. 1–17.
[13] Garg DP, Kumar M. Optimisation techniques applied to multiple manipulators for path planning and torque minimisation. Engng Appl Artif Intell 2002;15:241–52.
[14] Gen M, Chengh R, Oren SS. Network design techniques using adapted genetic algorithms. Adv Engng Software 2001;32:732–44.
[15] Goldberg DE. Genetic algorithms in search, optimisation and machine learning. Reading, MA, USA: Addison-Wesley; 1989.
[16] Hart PE, Nilsson NJ, Raphael B. Correction to 'A formal basis for the heuristic determination of minimum cost paths'. SIGART Newslett 1972;37:28–9.
[17] Hassoun MH, Sanghvi AJ. Fast computation of optimal paths in two- and higher-dimension maps. Neural Netw 1989;3:355–6.
[18] Health and Safety Executive, Safety of construction transport, A Survey of Standards and Accidents Associated with Construction Vehicle, UK. ; 2001.
[19] Holland J. Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press; 1975.
[20] Hurley S, Moutinho L. Approximate algorithms for marketing management problems. J Retailing Consumer Serv 1996;3(3): 145–54.
[21] Jung ES, Dohyung K. A man-machine interface model with improved visibility and reach functions. Comput Ind Engng 1996;30(3): 475–86.
[22] Kei LS, Srikanthan T. Dynamic multicast routing in VLSI. Comput Commun 2000;23:1055–63.
[23] Khoo LP, Ng TK. A genetic algorithm-based planning system for PCB component placement. Int J Prod Econ 1998;54:321–32.
[24] Kolen AW, Kan AH, Trienekens HW. Vehicle routing with time windows. Oper Res 1987;35(2):199–204.
[25] Lee S, Adams TM, Ryoo B. A fuzzy navigation system for mobile construction robots. Autom Constr 1997;6:97–107.
[26] Li H, Love PED. Genetic search for solving construction site-level unequal-area facility layout problems. Autom Constr 2000;9: 197–215.
[27] Li Z, Anson M, Li G. A procedure for quantitatively site layout alternatives. Constr Manage Econ 2001;19:459–67.
[28] Lobo FG, Goldberg DE, Pelkian M. Time complexity of genetic algorithms on exponentially scaled problems. IlliGAL Report No. 2000015. Illinois Genetic Algorithm Laboratory, University of Illinois at Urbana-Champaign; 2000.
[29] Mahjoubi L, Yang JL. An intelligent materials routing system on complex construction site. Logistics Inform Manage 2001;14(5/6): 337–43.

[30] Mercedes Gomez-Albarran M, Fernandez-Pampilon-Cestors AM, Sanchez-Perez JM. A routing strategy based on genetic algorithm. Microelectron J 1997;28(6–7):641–56.
[31] Modesti P, Sciomachen A. A utility measure for finding multi-objective shortest path in urban transportation networks. Eur J Oper Res 1998;111:495–508.
[32] Mustafa MA, Al-Bahar JF. Project risk assessment using the analytic hierarchy process. IEEE Trans Engng Manage 1991;38:46–52.
[33] Neapolitan R, Naimipour K. Foundations of algorithms, 2nd ed. Massachusetts: Jones and Bartlett Publishers; 1998.
[34] Perry JG, Hayes RW. Risk and its management in construction projects. Proceedings of Institution of Civil Engineers 1985;78(1): 499–521.
[35] Qu L, Xu G, Wang G. Optimisation of the measuring path on a coordinate measuring machine using genetic algorithms. Measurements 1998;23:159–70.
[36] Romeijn HE, Smith RL. Parallel algorithms for solving aggregated shortest path problems. Comput Oper Res 1999;26:941–53.
[37] Russell S, Norvig P. Artificial intelligence: a modern approach. New Jersey: Prentice-Hall; 1995.
[38] Sandurkar S, Chen W. GARPUS—genetic algorithms based pipe routing using tessellated objects. Comput Ind 1999;38:209–23.
[39] Schalkoff RJ. Artificial intelligence: an engineering approach. Singapore: McGraw-Hill; 1990.
[40] Seo KS, Lee YJ. Applicability of genetic algorithms to optimal evaluation of path predicates in object-oriented queries. Inform Process Lett 1996;58:123–8.
[41] Solka JL, Perry JC, Poellinger BR, Rogers GW. Fast computation of optimal paths using a parallel Dijkstra algorithm with embedded constraints. Neurocomputing 1995;8:195–212.
[42] Soltani AR, Tawfik H, Fernando T. A multi-criteria based path finding application for construction site layout. Sixth International Conference on Information Visualisation, London 2002;779–84.
[43] Sung K, Bell MGH, Seong M, Park S. Shortest paths in a network with time-dependent flow speeds. Eur J Oper Res 2000;121:23–39.
[44] Takaoka T. Shortest path algorithms for nearly acyclic directed graphs. Theor Comput Sci 1998;203:143–50.
[45] Tawfik H, Fernando T. A simulation environment for construction site planning. Fifth International Conference on Information Visualisation, London 2001;199–205.
[46] Tawfik H, Fernando T. A parallel genetic algorithm for optimising construction site layouts. Applied simulation and modelling. Proceedings of the IASTED International Conference, Spain 2001;250–4.
[47] Youssef H, Sadiq MS, Adiche H. Evolutionary algorithms, simulated annealing and tabu search: a comparative study. Engng Appl Artif Intell 2001;14:167–81.
[48] Yu G, Yang J. On the robust shortest path problem. Comput Oper Res 1998;25(6):457–68.