Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 15 (2012) 266 – 273

Procedia Computer Science

Virtual Worlds for Serious Applications (VS-GAMES'12)

Exploring the application of computer game theory to automated assembly

H. Yu[a], T. Lim[a]*, J. Ritchie[a], R. Sung[a], S. Louchart[a], I. A. St□nescu[b], I. Roceanu[b], S. de Freitas[c]

[a]*Heriot-Watt University, Riccarton, Edinburgh EH14 4AS, Scotland, UK*
[b]*National Defence University "Carol I", Bucharest , 50662, Romania*
[c]*The Serious Games Institute, Coventry, CV1 2TL, UK*

Abstract

This paper presents a method for automatic mechanical part assembly planning based on innovative use of artificial intelligence (AI) technology from games. The dynamic topological graph for the planning is generated using a heuristic search method based on characteristic information of each part. A prototype system has been developed to demonstrate the feasibility of the method. During the procedure, users are able to interact with the system due to the dynamic characteristics of the method.

Keywords: Virtual assembly; GOAP; Game theory; Assembly planning; Game AI; Serious game

1. Introduction

Over 40% of costs of an industrial product are associated to assembly operations [1]. A product may have numerous possible assembly sequences with each having a significant impact on assembly complexity and time. An optimal assembly sequence can translate to better configuration of equipment layout and use, reducing production time and cost. This information can also be used to improve or change the product's design.

Games and game theory has been investigated for various kinds of applications [2, 3]. Bellotti, Berta and de Gloria [4] explored gaming mechanisms for enhancing knowledge acquisition in virtual reality environment. Serious games on the other hand have played an important role in learning and training [5, 6]. In this paper, a method for automated assembly planning is presented, which borrows the idea of the AI theory used in

* Corresponding author. Tel.: +44-131-451-4353; fax: +44-131-451-3129.
E-mail address: t.lim@hw.ac.uk.

doi:10.1016/j.procs.2012.10.078

computer games. The hypothesis here is that assembly operations can be achieved through optimizing a dynamic topological graph using component information. The dynamic topological graph is generated in real time based on connection between the pair of parts through regressive search.

2. Related work

2.1. Assembly Planning

Assembly planning can be seen as a constrained procedure based on precedence constraint knowledge. Bourjault [7] proposed an approach for generating possible assembly sequences, which represented an early stage of exploration in automatic assembly sequence planning. Wolter [8] proposed a method using a constraint graph while Homem and Sanderson [9]explored a method using a relational model graph.

With the assistance of users' queries, interactive assembly can be achieved using connection information between the pair of components [10]. However, it is very difficult to generate the assembly sequence automatically. To overcome the automation difficulty, geometry-based methods have been implemented using feature reasoning or parametric information [9, 11-13]. These methods rely almost exclusively on heuristics and geometric feature identification, seldom taking into account of prior knowledge from assembly experts.

Nongeometric high level information such as expert knowledge or experience has also been considered in assembly planning. Swaminathan and Barber [14] presented a assembly sequence planning system based on previous experience, while Chakrabarty and Wolter [15] proposed a structure-oriented approach, which was used both as a framework for structure-dependent definitions of a good plan and as a tool to rapidly find good plans using expert advice. One of greatest challenges here is that it is difficult to capture expert experience for assembly [16]. This remains the case today.

The development of new technologies can enable designers and industrial engineers to speed up the decision making process. Virtual Reality (VR) technologies provide direct sensations for users to interact with a computer during the process of virtual assembly simulation [16-20]. The VADE system [19] uses constrained CAD models within a VR environment that represents the assembly area, and an expert human assembler can manipulate virtual parts and the assembly tools using both hands and dexterous finger tip-based manipulations to perform realistic assembly operations. The CODY Virtual Constructor [20] is a knowledge-based system that enables the interactive assembly of 3D visualized mechanical parts in a virtual environment. With the help of haptic system, users are able to feel physical forces and interact with virtual parts. Although researchers have proposed to integrate expert knowledge, a successful implementation has yet to surface. As the system becomes more complex with larger number of parts, generating a satisfactory assembly sequence is not trivial given the numerous permutations of associativity.

2.2. Computer Game Theory

As an alternative to the traditional AI planning technology in computer games, Goal-Oriented Action Planning (GOAP) proposed recently provides a more flexible behavioural strategy for non-player characters [21, 22]. GOAP is based on STRIPS (STanford Research Institute Problem Solver) [23] which is an automated planner including initial state and goal stages. GOAP is a decision-making architecture with the capability of searching the next actions [21, 22]. It allows agents to determine not only what to do, but how to do it in each step. Specifically, GOAP is defined in terms of goal, action, plan and formulate. A goal is any condition that an agent wants to achieve. An agent may have any number of goals. At any instant, once one goal is active, it determines the character's behaviour. A goal knows how to calculate its current relevance, and knows when it has been satisfied. The plan is simply the name for a sequence of actions. A plan that satisfies a goal refers to the valid sequence of actions that will take a character from some starting state to some state that satisfies the goal. An action is a single, atomic step within a plan that makes an agent do something. Each action knows when it is valid to run, and what it will do in the game world. In other words, an action knows its preconditions

and effects. Preconditions and effects provide a mechanism for chaining actions into a valid sequence. An agent uses a planner to formulate a sequence of actions that will satisfy some goal. In GOAP, each action has a static cost value calculated from the cost function, through which the planner can determine the next action.

A goal-directed character displays some measure of intelligence by autonomously deciding to activate the behaviour that will satisfy the most relevant goal at any instance. During the gaming procedure, a character that formulates its own plan to satisfy its goals exhibits less repetitive, predictable behaviour, and can adapt its actions to custom fit the current situation. In addition, the structured nature of GOAP architecture facilitates authoring, maintaining, and re-using behaviours.

3. Methodology and implementation

Many established assembly methods generate assembly sequences through knowledge reasoning, which either predefine possible plans or list constraints for the plan. The proposed method in this paper obtains the goal of forming an assembly sequence by satisfying the state in each step. To help calculate the cost the relationships of each pair of components is interpreted using logic matching knowledge. The optimal planning sequence is determined by the one with the least cost calculated from the cost function.

To establish whether GOAP could be a viable technique, a pump assembly was implemented to test the hypothesis and to evaluate its feasibility for virtual assembly. The goal is defined as the assembly of a pump comprising five components.

3.1. Component Action

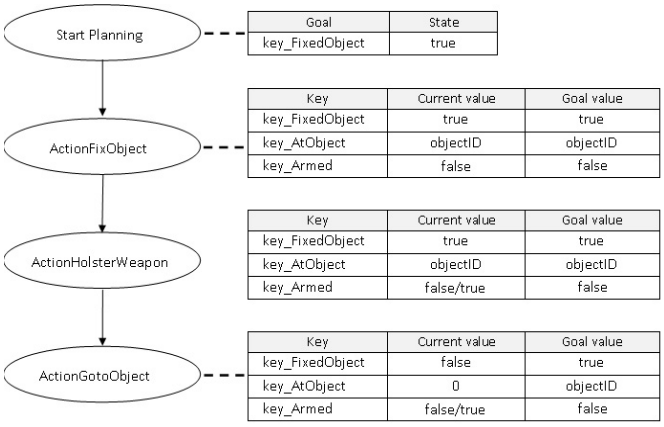


Fig. 1. Example of formulating a plan. ObjectID represents the ID of an object needing repair.

To demonstrate how GOAP is adapted for mechanical assembly, we first have a look at how GOAP works in a computer game. Consider the example given by the GOAP working group [24] - suppose the goal denoted by GoalRepairCar is repairing a car in the game while at the same time looking out for threats and so draw a weapon for defence. When, or if, the virtual avatar notices something is wrong with a car, GoalRepairCar returns "true" indicating that GoalRepairCar is active; otherwise it returns "false". Once GoalRepairCar is active, the virtual avatar will be in action to satisfy the WorldState. During this procedure, a sequence of steps was generated following the satisfaction of preconditions. Should a threat be detected, the avatar's goal would be drawing the holstered weapon in readiness for defensive/offensive mode. GoalRepairCar is deactivated once it is fully satisfied. However, if the avatar is aware that other objects require fixing, it reactivates the goal and plans again. Fig. 1 exemplifies this procedure and demonstrates the action planning strategy. Once the plan is

validated, it is then formulated as a sequence of actions to satisfy the goal. The sequence of actions to achieve the above goal is: ActionGotoObject | ActionHolsterWeapon | ActionFixObject.

For any mechanical assembly, a similar strategy could be employed to formulate an assembly sequence or strategy. The pump assembly as seen in Fig. 2 will be used as an example.

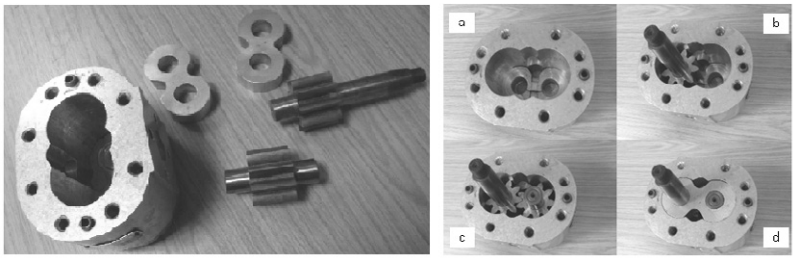


Fig. 2. (a) Pump components: housing; top plate; bottom plate; small cog; big cog. (b) Assembly sequence of the pump components.

Suppose the goal denoted by GoalAssemblyPump is assembling the pump. Once the goal is activated, the assembly procedure will start to search the state space in order to satisfy the WorldState. A sequence of actions is then formulated based on regressive search steps. As in games, occasionally, there may be more than one valid plan which satisfies the goal. The objective of the planner is to find only one best plan, thus an algorithm is applied to guide the search. Specifically, a cost function is associated with the actions to calculate the least costly sequence of actions. When applying GOAP in the context of mechanical assembly, one of the greatest challenges is selecting the optimal method for searching the action space.

The A* algorithm [25] for calculating the cost of the action search has been implemented. If taking the world state of the assembly as a node, we then use A* to calculate the heuristic distance from a node to the goal, where A* nodes represent GOAP world states and edges represent GOAP actions. After start, the algorithm proceeds to implement the plan with one action being taken at each step. Searching terminates when no further actions are needed. The action sequence with the lowest cost is preferable for the assembly. Fig. 2b illustrates the correct sequence of assembling the pump's five components.

3.2. Component Model Searching

GOAP adopts real time planning through regressive search in order to determine a set of proper actions to satisfy a predefined goal. The procedure starts the search with the goal and moves step by step based on the preconditions and effects of each stage. The search will attempt to find a valid plan that achieves the goal where possible, terminating only either when its preconditions has been met or cannot be met.

To implement the A^* algorithm for assembly searching, we propose to use two steps as follows:

- Roughly filter available components.

  The purpose of this step is to group all possible matching components. A property matching method is applied to roughly select all possible components which have matching properties with each other. For example, a 'component property' can include a threaded hole number and size index. Given a component with five threaded holes sized eighteen, the property matching algorithm then tries to find at least five threaded bolts with the similar thread size. If there are more than five bolts available, which have the same threaded size but different lengths, the algorithm will take all of them as the same group. A further calculation will be taken to choose the right ones in the next step.

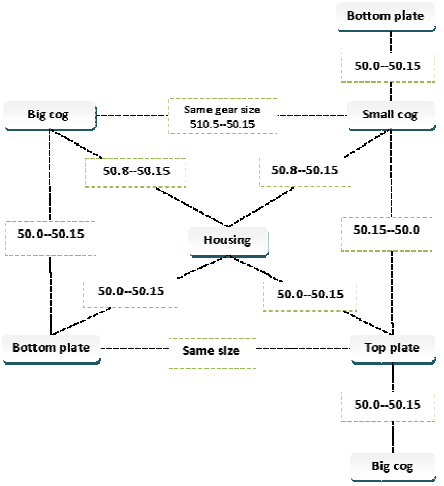
- A* searching algorithm is applied for all of these filtered components.

Table 1 shows a property table of five components of the pump in Fig. 2. In this simple example, we take properties such as external dimension, internal dimension, hole number, hole size, threaded hole number, threaded hole size and gear size. The matching properties of different components are marked by the same start markers.

Table 1. Pump component property.

Name	External dimension	Internal dimension	Hole number	Hole size	Thread hole number	Hole thread size	Gears
Housing	141.58, 140.74, 107.52	93.19, 54.2, 50.8*, 50.8*	2	50.8*	0	0	0
Big cog	147.62, 50.15*	22**	0	0	0	0	8.73, 8.1, 1.5****
Small cog	80.88, 50.15*	22.12**	0	0	0	0	8.73, 8.1, 1.5****
Top plate	93.0***, 50.0*	22.3**	2	22.3**	0	0	0
Bottom plate	93.0***, 50.0*	22.3**	2	22.3**	0	0	0

The property matching can happen between any pair of, or more, components based on properties in Table 1. Fig. 3 shows the matching network of the five pump components. From the property matching network, we can see that the big cog and the small cog match because they have the same gear size including some internal and external dimensions. The housing, top plate and bottom plate match due to some of the internal dimensions of the housing matching some external dimension of the top plate and bottom plate.



(a)

Goal	key_assemblePump = true

Component	Current State	Goal State
Housing	key_assembleHousing = false	true
Top plate	key_assembleTopPlate = false	true
Bottom plate	key_assembleBottomPlate = false	true
Big cog	key_assembleBigCog = false	true
Small cog	key_assembleSmallCog = false	true

Action	AssembleHousing	AssembleTopPlate
Preconditions	key_assembleBottomPlate = true	key_assembleBigCog = true or key_assembleSmallCog = true
Effects	key_assembleHousing = true	key_assembleTopPlate = true

Action	AssembleBottomPlate	AssembleBigCog
Preconditions	key_assembleHousing = true	key_assembleSmallCog = true or key_assembleTopPlate = true
Effects	key_assembleBottomPlate = true	key_assembleBigCog = true

Action	AssmebleSmallCog
Preconditions	key_assembleBigCog = true or key_assembleTopPlate = true
Effects	key_assembleSmallCog = true

(b)

Fig. 3. (a) Property Matching Network for the Pump assembly. (b) Implemented GOAP definitions for the pump.

After property matching, the next stage is to use a search algorithm to validate connections. Here we describe how the AI planning is implemented for the pump assembly based on the GOAP AI in computer game. Fig. 3b shows some definitions of the implementation. The goal of the assembly is denoted by key_assemblePump. The current state of key_AssemblePump becomes "true" once the goal is achieved. To achieve the goal, there are five actions along with the components including AssembleHousing, AssembleTopPlate, AssembleBottomPlate, AssembleBigCog, AssembleSmallCog. As with actions in computer game, the action of each component includes preconditions and effects. Preconditions of the action are properly set beforehand. Each action is associated with a cost, which will be used by A* search. Those preconditions

play an important role in the search, which validate certain constrains for the action to carry on based on the planner.

To commence the assembly process, preconditions and effects for each component has to be defined. These are input manually and are based on expert knowledge. With more components incorporated in the system, it will become more easily to use and the preparation for defining the component will significantly be reduced. There are some exceptions for some actions, which do not have any preconditions for action implementation, for example the housing component in the pump example. Fig. 4 shows the procedure of the GOAP architecture for planning in a design for assembly (DFA) operation. The goals are specific, based on the design requirements for the pump function and assembly requirements.

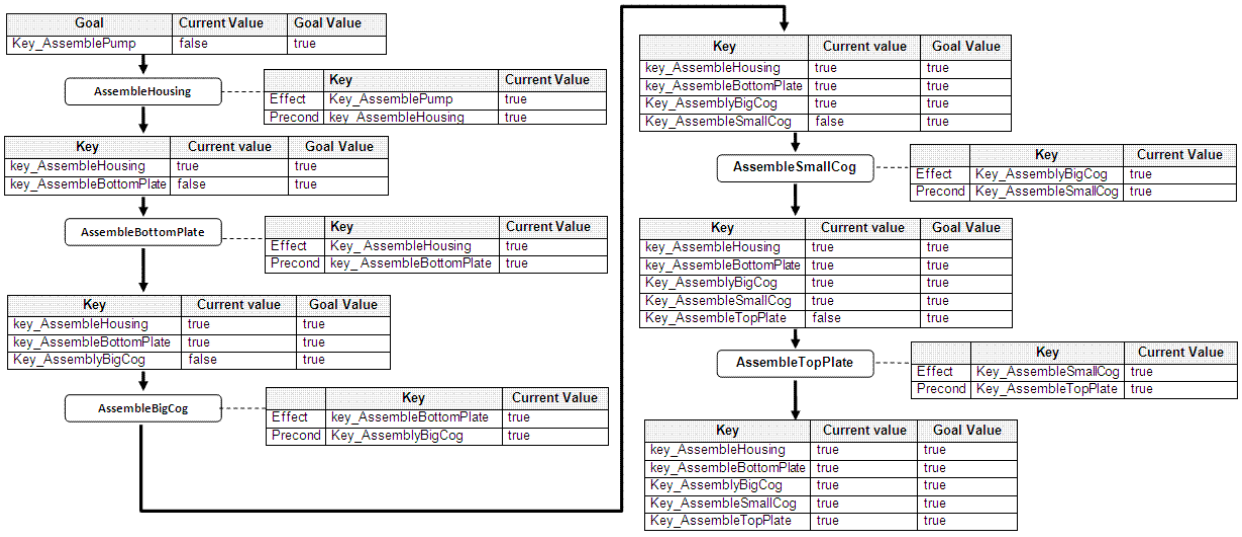
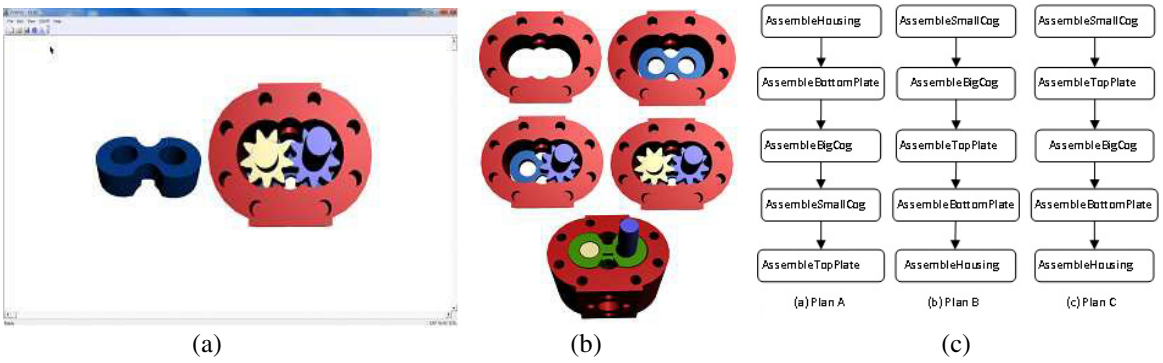


Fig. 4. GOAP architecture for design for assembly (DFA) planning.

4. Results and discussion

The interface of the prototype GOAP planning system in Fig. 5a is coded in C++ and implements the OpenGL renderer. The concept was initially tested using the physical components then with virtual models. Fig. 5b shows the virtual assembly sequence. These were uploaded into the planner in *.stl format.



(a) (b) (c)

Fig. 5. Generating assembly plans using GOAP. (a) The user interface. (b) Assembly sequence of virtual pump components. (c) GOAP generated assembly plans. All plans generated are tangible with Plan A deem optimal.

An assembly sequence of the virtual models using the definitions as illustrated in Fig. 3b and the governing architecture as shown in Fig. 4. The assembly sequence in Fig. 5b corresponds to the optimal plan as generated by the GOAP output in Fig. 5c. Plan A is where the assembly sequence starts with the housing component. This is the optimal solution, which comes from experience of professional assembly expert. In this case, the cost of this assembly is minimal. Plan B and Plan C start with the small cog. Assuming that Plan A has been generated and the user changes his/her mind during the assembly by choosing the top plate before small cog, the GOAP will re-plan the assembly. Since the GOAP system detects the change it forces the system to search again to satisfy the goal. Therefore, a new assembly sequence is generated accordingly as in Plan C in Fig. 5c.

The time taken to generate these plans was less than 0.3 seconds. This speed is attributed to the low number of components and it can be assumed that more complex assembly models will incur greater processing with increase in time to deliver sane solutions. While still in its early stage, the results have thus far been encouraging. Collision detection, physics simulation and additional behavioural algorithms have not yet been integrated.

5. Conclusion

Assembly sequence planning plays an important role for product design and assembly automation. A common method for automatic assembly planning relies on the reversing process, i.e. disassembly. Even though it is a reversible process, it still depends on expert knowledge for obtaining the disassembly information. The challenge for assembly planning lies in determining the optimum sequence.

An assembly procedure can start with any component but if starting with a *too-general* and/or *non-critical component*, it may not lead to a valid assembly sequence. In the GOAP example, two possible sequences can be made:

Plan A: "AssembleHousing | AssemblyBottomPlate | AssembleBigCog | AssembleSmallCog | AssembleTopPlate"

Plan B: "AssembleHousing | AssemblyBottomPlate | AssembleBigCog | AssembleTopPlate | AssembleSmallCog"

The difference lies on the order of AssembleBigCog, AssembleTopPlate and AssembleSmallCog. In this case, Plan B would be invalid since the gears would not be able to mesh. As always, the final plan is determined by the cost of actions and preconditions. Therefore, it is recommended to start with a relatively critical component, i.e. components that are usually not a standard or common part. For example, a bolt or nut would not be considered as an important component to start with, since they are too general and could be assembled on any part. In the pump example, all five parts can be viewed as critical components. Since there is no uniform rule for definition of "critical component", we generally define the "critical component" based on experience.

The GOAP architecture appears to have a clear advantage over some common techniques such as finite state machines (FSM). One of the important features of GOAP lies on its dynamic replanning capability. The planner constantly checks the component's current plan for its validity. If anything changes, the GOAP system will replan and choose different actions to meet the established goal. The dynamic replanning feature enables the system to change the assembly sequence according to the selection of components by the user. This dynamic feature of GOAP makes it more flexible and superior than the FSM technique.

In this paper we presented an approach for applying computer game theory, specifically behavioural AI methods, to mechanical assembly. We have demonstrated the feasibility of adapting the dynamic planning method GOAP to assembling a pump model. The system makes the learning procedure more pleasant and gets more engagement from beginners or students.

In the future, we will apply this method to more complex objects with more parts with higher complexity in terms of part shape. Applying the method to significantly more parts will introduce additional challenges, therefore an improvement to the property definition method and ways for property search will be required. The current limitation here pertains to the simplicity of the test components and limited interactivity of the system.

To make this method more usable, it is anticipated that the GOAP module is transferred to an existing design environment; Blender could provide an interface to incorporate new packages and physical simulation.

Acknowledgements

This project is partially funded under the European Community Seventh Framework Programme (FP7/2007 2013), Grant Agreement nr. 258169 and EPSRC/IMRC grants 113946 and 112430.

References

[1] Shan HB, Li SX, Gong DG, Lou P (2006) "Genetic simulated annealing algorithm-based assembly sequence planning,". International Technology and Innovation Conference 2006 (ITIC 2006), CP 524:1573–1579.

[2] Hernandez G, Seepersad CC, Mistree F. Designing for Maintenance: A Game Theoretic Approach. Eng. Opti, 2002; 34(6):561-577.

[3] Louchart S, Lim T, Al Sulaiman H., 2009. "Why are Video-Games relevant test beds for studying interactivity for Engineers?" International Simulation and Gaming Association Conference. Singapore, paper O-25#4.

[4] Bellotti F, Berta R, De Gloria A, Zappi V, 2008. "Exploring gaming mechanisms to enhance knowledge acquisition in virtual worlds," 3rd International Conference on Digital Interactive Media in Entertainment and Arts, p. 77-84 .

[5] Neill, T. Serious games: learning for the igeneration. J. Development and learning in organizations 2009; 23(4):12-15.

[6] Petridis P, Nadim W, Bowden S, Goulding J, Alshawi M, 2009. "ManuBuild Construction Site Training Simulator for Offsite Manufacturing," IEEE 1st International Conference for Serious Application, Coventry, UK, p. 170-173.

[7] Bourjault A, 1987. "Methodology of assembly automation: A new approach," 2nd Int'l Conf. of Robotics and Factories of the Future, San Diego, California, p. 37-45.

[8] Wolter JD. On the Automatic Generation of Plans for Mechanical Assembly. PhD thesis, U. of Michigan, Ann Arbor, MI, 1998.

[9] Homem de Mello LS, Sanderson AC, 1989. "A correct and complete algorithm for the generation of mechanical assembly sequences," IEEE International Conference on Robotics and Automation, 1:56-61.

[10] Wilson, RH, 1996. "A framework for geometric reasoning about tools in assembly," IEEE International Conference on Robotics and Automation, 2:1837-1844.

[11] Baldwin DF, Abell TE, Lui MC, De Fazio TL, Whitney DE. An integrated computer aid for generating and evaluating assembly planning. IEEE Trans. Robot Autom., 1991; 7(1):78–94.

[12] Ataie-Ashtiani B, Lockington DA, Volker RE, Eng TH, Ling ZK, Olson W, McLean C. Feature-based assembly modeling and sequence generation. Comput. Ind. Eng., 1997; 36(1):17–33.

[13] Lee S. Subassembly identification and evaluation for assembly planning. IEEE Trans. Syst. Man. Cybern., 1994. 24(3):.493–503.

[14] Swaminathan A, Barber KS. An experience-based assembly sequence planner for mechanical assemblies. IEEE Trans. Robot Autom., 1996; 12(2):252–266.

[15] Chakrabarty S, Wolter J. A structure-oriented approach to assembly sequence planning. IEEE Trans. Robot Autom., 1997; 13(1):14–29.

[16] Ritchie JM, Simmons J.EL, Carpenter ID, Dewar RG., 1995. "Using Virtual Reality for Knowledge Elicitation in a Mechanical Assembly Planning Environment," Proc. of 12th Conf. of the Irish Manufacturing Committee, Cork, Ireland, p. 1037-1044.

[17] Jung B, Hoffhenke M, Wachsmuch I, 1997. "Virtual Assembly with Construction Kits," ASME Design Engineering Technical Conferences, Sacramento, California, paper DETC97/DFM-4363.

[18] Lim T, Ritchie JM, Dewar RG, Corney JR, Wilkinson P, Calis M, Desmulliez M and Fang JJ. Factors affecting user performance in haptic assembly. Virtual Reality, 2007; 11(4):241-252.

[19] Jayaram S, Jayaram U, Wang Y and Tirumali H. VADE: A Virtual Assembly Design Environment. Computer Graphics and Application, 1999; 19(6):44-50.

[20] De Fazio TL, Whitney DE. Simplified Generation Of All Mechanical Assembly Sequences. IEEE Trans. Robot Autom., 1987; 3(6):640–658.

[21] Orkin J. Applying Goal-Oriented Action Planning to Games, AI Game Programming Wisdom 2. Charles River Media; 2003.

[22] Orkin J. "Three States and a Plan: The A.I. of F.E.A.R," Game Developer's Conference., San Jose, CA, USA, 2006.

[23] Fikes RE, Nilsson NJ. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. Artificial Intelligence, 1971; 2:189-208.

[24] GOAP Working Group, http://www.igda.org/aireport/2004/goals

[25] Hart PE, Nilsson NJ, Raphael B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Man. Cybern.*, 1968; **4**(2):100-107.