Self-programming Construction Machines

Robert L. Winter* and Jonathan O'Brien*
* School of Civil Engineering, University of New South Wales, Sydney 2052, Australia

ABSTRACT

Self-programming, intelligent, flexible machine systems offer a desirable alternative to high-volume, hard automation solutions in the complex, ill-structured and dynamic field of construction. A theory of construction is developed which shows that the nature of construction requires the capability for problem solving. Highly complex, dynamic problems require a high degree of intelligence, or problem-solving flux, for solutions to be found. An opportunity exists to apply machine intelligence capable of autonomously attacking construction work. Economic viability underpins the design problem, and the specifications developed in this paper for an effective solution show that self-programming systems are preferable for the low-repetition environment of construction. An architecture for a self-programming system is developed, showing application and multi-agent potential, and highlighting the need for such systems to be developed.

1. INTRODUCTION

The synthesis of technically and commercially effective robotic construction agents is an engineering problem that is far from having been solved. Great strides have been made in the field of factory-type deterministic robotics, but most of the current industrial robotics work does not address the needs of the building and construction sectors where unstructured environments dominate. For technical reasons, building and construction robotics requires a range of devices that are much 'smarter' than those currently used in the manufacturing industry. Such machines or devices do not as yet exist. Furthermore, recent remarks from the managements of the existing half-dozen or so large industrial robotics multi-nationals makes it clear that there is no interest from the mainstream robotics community to invest in the development of the next generation of versatile, field robotic machines (Demark 1995)

The 'smartness' and technical capability issues related to engineering solutions require an understanding of the nature of construction activity. One part of this paper focuses on the major intellectual challenge of developing smart, or high IQ field robots. The economics of any technically effective set of robotic agents, however, is intrinsically connected to the cost of programming them. Automated systems that require "human-in-the-loop" programming or set-up are generally uneconomic for one-off or small job runs, which are pervasive in construction work at present. Large-volume automated systems offer a solution to problems where the environment may be constrained sufficiently to simplify operations and where the resulting high set-up cost can be amortised over a large number of iterations. The major focus of this paper is to show that self-programming machines offer a preferable solution.

2. A THEORY OF CONSTRUCTION

2.1 THE PROBLEM-SOLVING NATURE OF CONSTRUCTION

Construction is an activity undertaken to fulfil some specific human need or desire. It constitutes a problem, using a broad definition (Krick 1969), in that there is a desire to transform the environment from its current state to a more desirable one. A more pointed definition attributed to Newell and Simon (1972), states that a problem exists when there is a desire for a change of state, but a solution (the required series of actions) is unknown. For our purposes, it is assumed herein that the environment, modelled as a system, has many variables and that the interaction between construction systems and their environment is complex. In fact there are an infinite number of variables (Ashby 1960), but we choose to deal with only a representative number. For the many cases where the solution is non-trivial, a system capable of undertaking construction work thus requires a problem solver. O'Brien and Compton (1987) describe the processes used by human workers, and the following are required by any system, with respect to a problem, for it be considered **autonomous**:

1. Information analysis and gathering
2. Problem solving, reasoning and design
3. Physical process control and management
4. Physical action or use of effectors .

The problem-solving stage may be further subdivided (Krick 1969) into:

(i) A search for alternative solutions
(ii) An evaluation of alternatives
(iii) The selection of the preferred solution

The problem solver in its environment also conforms to the model of an *Information Processing System* described by Newell and Simon (1972) as shown in Figure 1.

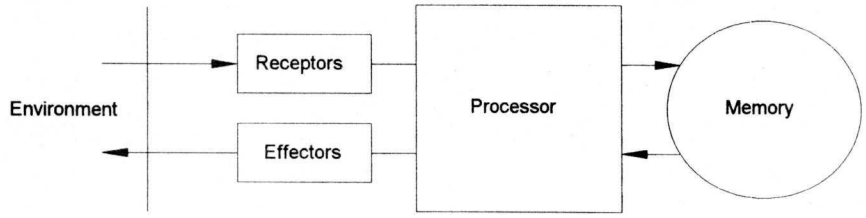

Figure 1 Information Processing System Model (Newell & Simon 1972)

The "problem", in construction, consists of transforming the environmental variables from one state to another. In turn, this constitutes the discovery of a set of basic processes that will effect the transition. Thus the architecture of a problem solving system may also be represented as a combination of a programmer, which finds a solution consisting of a program to execute a set of basis processes, and an underlying programmable system in which these processes will achieve the goal state (the programmer knows the transfer functions of the programmable system), as shown in Figure 2.

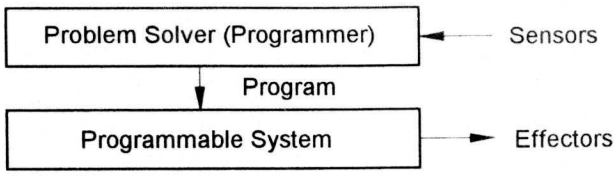

Figure 2 A Programming-based Representation of a Problem Solving System

2.2 SEARCH IN PROBLEM SOLVING

Planning or problem solving may be considered to be a search (Winstanley 1991). Search size is related to the number of environmental variables, and their range of values, as well as to the nature of the task, which together constitutes the **task environment**. The task environment is represented by the problem solver as a **problem space**. Problem **difficulty** relates to the size of the problem space (Newell & Simon 1972). The combinatorially explosive nature of search related to the difficulty of planning is noted by Beer (1995) and, for a 'pure' search, is proportional to the search space size. For a search tree of breadth b and depth d, the size of the search space is of the order of b^d (Ginsberg 1993).

The ability to find solutions depends on the search capabilities of the problem solver. A greater search capability, in terms of a static capacity for covering the 'field' which constitutes the problem space means a higher probability of finding a useful solution. Search capability can be enhanced by **knowledge** and heuristics, which prune the search space, leading to faster solutions (Selvestrel 1995; Ginsberg 1993). Since the problems of artificial intelligence are characterised by search and knowledge (Ginsberg 1993; Winstanley 1991), a specific definition of intelligence may be the capacity for searching problem spaces.

Machine intelligence, a preferred term to artificial intelligence (Lee 1989) is a moving benchmark (Zadeh 1996). What was considered high intelligence a few years ago is now considered low in intelligence. What is needed is a match between the complexity of the problem space and the intelligence of the system.

2.3 THE TIME SERIES OF PROBLEMS - PROBLEM DYNAMICS

Section 2.2 examined a single problem in isolation. If the motive is to build useful construction machines, then it is advantageous to consider a major construction project to be hierarchically decomposable into a series of sub-problems. Assuming the project to have a somewhat serial nature, the sub-problems may be considered a **time series** of problems. This is true for any level of decomposition, and thus major projects may also be considered a time series. Thus a problem-solving construction system must solve many problems in a series.

If each sub-problem constitutes a search for solutions, then it is important to consider the time frame in which the problems need to be solved. The problems may also change over time. The number of problems which need to be solved in a given time frame, and the rate of change of the problem parameters, (a frequency), constitute what may be termed the **problem dynamics.** The difficulty of solving a time-series problem may be measured by a two-dimensional criterion, a product of the problem complexity (K) and problem dynamics (1/T), as shown in Figure 3.The suitability of a machine for solving a time-series of problems may be determined by a corresponding product, i.e. the search capability (S) multiplied by the solution frequency (1/t). This is termed **Problem Solving Flux**, a more generalised definition of **machine intelligence**.

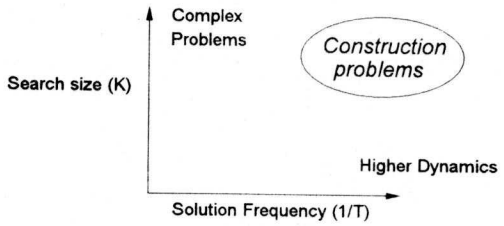

Figure 3 The Difficulty of Solving a Time-Series of Problems

2.4 UTILITARIAN MEASURES FOR SOLUTION EVALUATION

The selection of a solution requires a measure of preference, value, or utility to evaluate different alternatives. Furthermore, it is assumed here that a rational designer will only accept a solution which is **economically viable**. The viability depends on the perceived or measurable value of an outcome, as well as the actions required to achieve that outcome, which have real costs (Lee 1989). To describe this, a solution may be termed **viable** if:

$$\Delta U > 0, \qquad \text{where} \quad \Delta U = B - \sum_{i=1}^{n} C_i,$$

B is the utility derived from the outcome and the values of C are the costs of the incremental units of action required to achieve the outcome. A rational designer will also try to maximise ΔU. Thus the solution boundaries are not only set by technical feasibility, but also by economic viability, which in itself is a movable boundary.

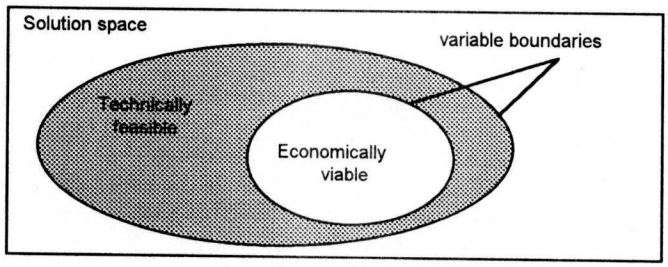

Figure 4 Field of Solutions Constrained by the Need for Economic Viability

3. SOLUTION POSSIBILITIES

3.1 AUTONOMY CONSIDERATIONS

For autonomous operation, a system must have adequate sensory, intelligence and action/effector capacity (Rosheim 1994) to solve a problem and carry out a solution to achieve a goal. Currently, construction systems often consist of a hybrid system, employing humans to undertake parts of the process. A typical distribution of roles is shown in Figure 5.

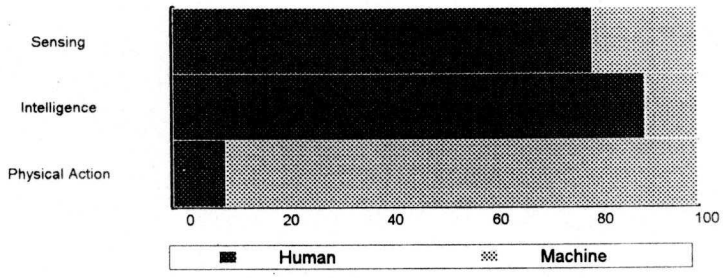

Figure 5 The Human Content of a Typical Construction Automation System

3.2 VIABILITY CONSIDERATIONS

Relative to a single problem, the effectiveness of a system in terms of economic viability depends on the total cost of the problem solution. A human is preferable to a machine for problem solving if it is cheaper for the same task. Thus systems to date have used machinery for the physical parts of the process, because of their high cost efficiency, have used limited sensors because of their high expense, and virtually no artificial intelligence for problem solving, because problems have been so complex, even for 'simple' tasks. In space operations, where the cost of human labour is relatively very high, autonomous machines for activities including construction are required to ensure the affordability of many missions (Heer 1988). If machine-only systems can be engineered that will out-perform humans for construction purposes, their applicability will definitely increase.

3.3. FLEXIBILITY CONSIDERATIONS

(A) Algorithmic Solutions

For machines which are very limited in terms of their information gathering capabilities and problem-solving flux, one approach is to use machine-only systems only in situations where the problems are very simple. In hard automation, the problem solving capabilities are removed from the machine and the system is applied only to problem spaces in which the solution has already been formulated as an **algorithm**. The machine does no problem solving and is thus unintelligent. In this situation, a large amount of effort is devoted to planning, but there is little applicability to a changing environment. Whilst inflexible, a large initial investment is justified for large numbers of repetitions of the same problem, and may be cheaper than a human-based, flexible solution (Figure 6).

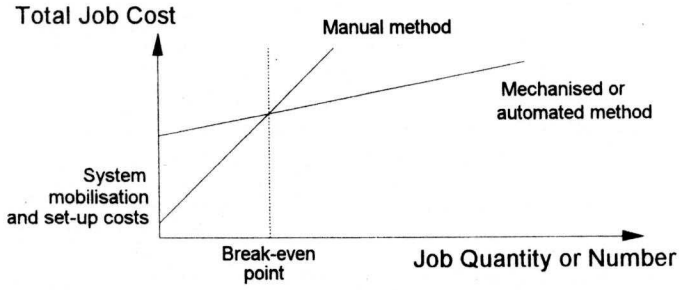

Figure 6 A Typical Cost Model of Technology Competition

The greatest problem in hard automation is that of flexibility, which relates to the ability to be economically redeployed as the boundaries of a problem change. The assumptions made about the scope of problems, and the technical and economic environments, e.g. that people will accept prefabricated boxes for homes for the next ten years, may lead to economic disaster when these boundaries change and the initial investment can no longer be amortised. This is analogous to having spent a lot of resources building a hammer and then finding no nails to hit. Inflexibility relates primarily to the **cost of finding and implementing a solution to a new problem**. An improvement on the hard automation strategy is to provide a programmable sub-layer which can be used for a number of different problems, e.g. an industrial robot, but again a human must provide the program, which is determined **off-line**.

Application examples of **playback machines** include the well-known Shimizu SSR-2 Spraying Robot and the Putzmeister programmable concrete pump. In a higher class of capability are machines which utilise model-based control, along with run-time data to allow for environmental imprecision, e.g. Trevelyan's sheep-shearing robot (Trevelyan 1992). Algorithmic machines with programming aids can reduce the scope of the problem, e.g. the Aachan robot system which programs robots directly from CAD models (Galya 1994).

(B) Real-time programming

Another type of system is one which consists (refer figure 2) of a highly programmable layer, which can be quickly reconfigured to undertake multiple tasks. The solution space of such a system is thus very large, but requires a very intelligent problem solver to control it, and the nature of the problem is such that the program must be provided **on-line**, in real time. This is the situation which exists with a human-controlled machine such as an excavator. The machine has many degrees of freedom, but requires the intelligence of a human controller to provide the programming and sensing capabilities. A key feature of such systems is that they offer generic capabilities, which allows them to be produced in high volumes and at low cost compared to hard-automation systems.

(C) The Self-Programming Solution

With increasing amounts of computational power, the thinking parts of the problem are becoming more affordable, thus allowing systems which use AI to provide flexibility by allowing low-cost methods of solving problems. Highly programmable physical systems, such as robot arms, can be coupled with an intelligent programmer to produce a flexible solution to construction problems. The implementation is immaterial as one could have real-time expert systems, neural networks, or any other feasible machine problem-solving capacity.

At present, however, intelligent machine systems do not have the problem solving flux required to autonomously operate in the current construction environment with the performance of a human, or a human-assisted system. As it stands, construction activity requires complex, real-time capabilities. To allow a transition from current, human-based systems, an effective plan for implementation would involve the successive mechanisation of the thought processes required for the human, by progressively layering intelligence over low-level functions of programmable machines (such as in Internet protocols) to make the programming required to achieve a task less computationally intensive. At a critical point, this layering will reduce the problem space sufficiently to allow the available intelligence capabilities of computers to be able to assume control. (A bottom-up approach). At the same time, simple programmable tasks can be carried out immediately, while maintaining a

system architecture that can be cheaply and quickly extended (A top-down approach). To provide a pervasive solution, this two approaches must meet. Examples of systems which have used the self-programming approach are the TRC Helpmate , a hospital servant which uses AI to navigate the corridors of a hospital (Rosheim 1994), and the problem-solving, soma-cube-world assembly robot system described by Petropoulakis and Malcolm (1990).

A definite engineering opportunity for building self-programming systems exists, underpinned by the need to increase the automated content of construction. The next question to be addressed is how to build such a system.

4. A PROPOSED ARCHITECTURE FOR SELF-PROGRAMMING CONSTRUCTION MACHINES

Based on the two-layered self-programming model proposed in Figure 2, this section proposes a specific system architecture whose design aims to incorporate the features of **autonomy**, **viability** and **flexibility**.

4.1 DESCRIPTION OF A SELF-PROGRAMMING SYSTEM

With reference to Figure 7, the system consists of two main parts. It consists of an intelligent programming layer, which has the function of sensing the environment, forming goals, determining a plan or program for action, and then sending this program to the programmable sub-layer which is designed to execute the plan.

Rather than being an intelligent 'black box', the intelligent programmer (IP) is modularised to reflect the specific aims of the system. In order to satisfy the viability criteria of the user, the intelligent programming layer is equipped with a specialised pre-processing unit, named the **value set**, which contains a set of transforms for mapping the values of the vector of input parameters to an evaluation vector to be used by the action engine in the planning stage. The value set is further divided into a **benefit set**, in which the transforms relate specifically to goal states, and a **cost set**, in which the transforms relate to problem parameters.

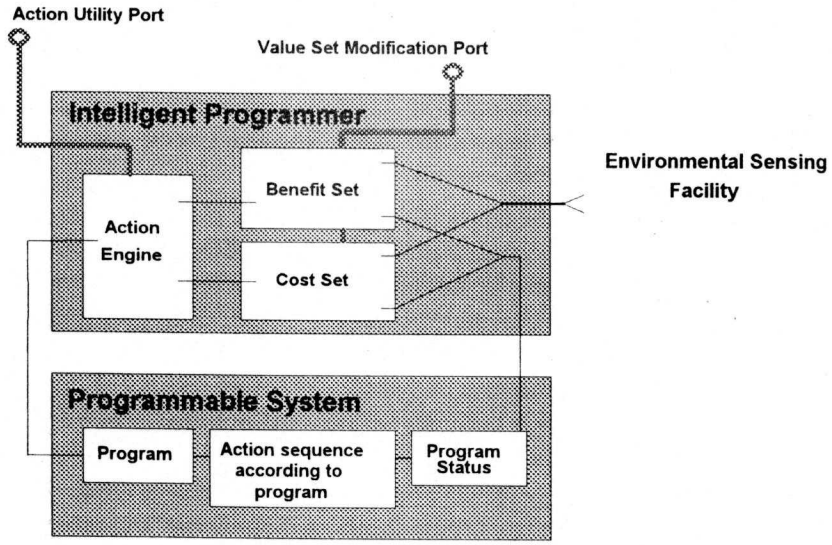

Figure 7 Self-Programming System Architecture

The **action engine**, coupled to a sensory facility and value set, drives the system. The information which it receives is used to determine potential goal states, which result from the functional purpose of the system. It then derives, through problem solving, and the cost set, whether the job is viable in a utilitarian sense. It thus has the feature of being **self-initiating**, since the IP, when in a suitable environment, will cause the programmable system (PS) to begin action. It may also remain in a latent state, as conditions may not be suitable for action. For example, a programmable truck's IP may have identified a point on the other side of a field as a potential goal, but weather conditions may make the crossing too costly to be of use, and therefore the vehicle's IP may search for other paths or wait until conditions are favourable (i.e. viable).

The architecture described possesses flexibility on three levels. Firstly, assuming that the machine transfer functions are available for the PS's operation, the IP can be easily adapted to systems of the same generic function. For example, the IP may be quickly installed in several different makes of truck, assuming that they are functionally equivalent. Secondly, assuming an appropriate capability for sensing, the number of transforms, (which may also be considered to be **weighting functions**) in the value set, may be increased to reflect an increased need for refinement in a solution. As an example, the paths travelled by our vehicle may depend on the surface roughness (and hence the cost of tyre wear), but we may now wish to extend the analysis involved in the problem-solving process to include the grade of the terrain, in which case a software-based transform would be added to the value set to include grade of the surface. Thirdly, the IP is equipped with a value set modification port which, with a feature similar to Direct Memory Access on a PC, the values of the transforms may be directly modified. Thus in continuation of the programmable truck example, a new type of tyre may be fitted, with lower wear characteristics, thus requiring a modification of the transform relating to tyre wear, by reducing the cost related to navigation.

With regard to the second level of flexibility, namely that of adding transforms, the system experiences only a *linear* increase in the computation required, as each transform is used only for the evaluative part of the problem solving process, thus making the system easily extendible to suit growing knowledge of environmental effects on the solution. The following section presents application examples to illustrate this point and also to demonstrate the real-world potential of such systems.

4.2 APPLICATION EXAMPLES

(A) Autonomous Truck

Continuing the illustration of the above section, an autonomous truck may have as its purpose the haulage of material across a field to a dump site. Part of this problem, which is now the focus of attention, is to select an appropriate dump point, and an appropriate path. The field, including the dump area, may be represented as a series of tiles (Fig. 8). To simplify the problem, the tiles at the furthest edge of the tiled plane are potential targets for dumping. The value of the 'dump' at each tile is a function (not necessarily linear) of the surface depth. The function is stored in the value set. In order to solve the problem, the PS must determine a connected sequence of tiles to reach any of the dump sites that maximises the total utility of the exercise or, at least, finds a solution in the time provided which is viable. Crossing each tile has a particular cost which depends on the values of a number of characteristic variables associated with the particular tile. These variables are the transform functions in the cost set. The effect of the preprocessing the parameters is to map the constraints into a 'cost space'.

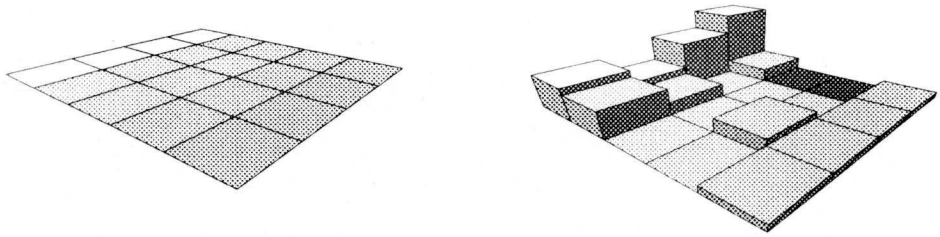

Figure 8 Tiled Representations of the Problem Field for an Autonomous Truck

In this case, the most desirable solution is that which maximises the total net utility between the starting point and the finishing point. From a flexibility point of view, such a system can be recalibrated to suit the required conditions by changing the weightings, to allow for such priorities as fuel consumption or wear, with a direct reflection in the problem solving process.

(B) Concrete Finishing Robot

A programmable trowelling robot could be coupled with an IP in a similar manner, with the robot being attracted to unfinished areas on the basis of a smoothness function. The degree of finish could be controlled simply by increasing the goal value for smoothness.

4.3 MULTI-AGENT SYSTEMS

To maximise the flexibility and utility of intelligent systems, a multi-agent scenario provides the opportunity to distribute the intelligence required for a major problem to a number of smaller units of intelligence, as well as to distribute the physical action. We may deploy a small fleet of intelligent trucks to haul large amounts of materials, rather than use one large truck, in order to increase redundancy and to be able to vary fleet size to match material flow rates. A multi-agent system, however, creates the potential for problem environments which are altered by the action of other agents on the environment, for conflict between the agents, and for problems which require agents to coordinate their actions in order to solve their problems. For example, a dumping-spreading combination, consisting of an off-road truck and a dozer, does not need to be coordinated to a great extent, but the operation of the dozer is such that it is dependent on a supply of materials from the truck. Without any communication between the two units, the waiting behaviour of an autonomous dozer controlled by an IP is such that the existence of a heap of material might be used to couple the two systems. This may be considered a 'focal point' for coordinating two systems (Fenster et al. 1995) without explicit communication. The key point here is that by linking systems through an intermediary environmental state, the loose coupling of systems allows flexibility in that any system which is functionally equivalent to the haul unit is can be used.

The subject of conflict resolution and communicative coordination is not considered here, but it is suggested that, for such a purpose, communication based on a comparison of goal values and intended plan costs may form an efficient language for solving multi-agent problems of this type (Winter et al. 1995). The **action utility port** is included in the IP described above as a port for communicating intended goals and values to other systems.

5. CONCLUSION

Construction is an activity which involves problem solving. Any system capable of construction must include intelligence of a sufficient problem-solving-flux capacity to match

the complexity and dynamics of such problems. However, for the type of problems most common in construction, machines to date require humans to provide the intelligence for adequate flexibility. As a competitor, an autonomous construction machine must therefore possess a high level of machine intelligence to be economically viable. Self-programming systems provide the potential for flexibility, autonomy and industrial viability. The engineering design problem is thus to build low-cost, self-programming machines. This is a very difficult problem. The contribution of this paper is that an architecture for a suitable machine has been demonstrated, indicating the feasibility of developing autonomous agents and systems of autonomous agents suited to the real world of construction.

6. REFERENCES

Ashby, W. R. 1960, *Design for a Brain*, 2nd ed., Chapman and Hall Ltd, London.

Beer, R. D. 1995, 'A dynamical systems perspective on agent-environment interaction', *Artificial Intelligence*, vol. 72, pp. 173-215.

Demark, S. 1995, Chief Executive Officer, ABB Robotics International, Keynote address 26th ISIR, Singapore 4-6 Oct.

Fenster, M., Kraus, S. & Rosenschein J. S. 1995, 'Coordination without Communication: Experimental Validation of Focal Point Techniques', in *Proc.First Int.Conference on Multi-Agent Systems*, AAAI Press, Menlo Park, California, 12-14 June, pp. 102-108.

Galya, B. 1994, 'Universal robot planning system fôr small batch manufacturing of products with large dimension and several thousands of spot and arc welds', APS Mechatronik, Europaisches Centrum fur Mechatronik Aachan, Research data sheet, April.

Ginsberg, M. L. 1993, *Essentials of Artificial Intelligence*, Morgan Kaufmann, California.

Heer, E. 1988, 'Toward Intelligent Robot Systems in Aerospace', in *Machine Intelligence and Autonomy for Aerospace Systems*, eds E. Heer & H. Lum, American Institute of Aeronautics and Astronautics Inc., Washington D.C.

Krick, E. V. 1969, *An Introduction to Engineering and Engineering Design*, 2nd ed., Wiley, N.Y.

Lee, M. H. 1989, *Intelligent Robotics*, Open University Press, Milton Keynes, Great Britain.

Newell, A. & Simon, H. A. 1972, *Human Problem Solving*, Prentice-Hall Inc., New Jersey.

O'Brien, J. B. & Compton, R. 1987, 'Field Studies of the Cognitive Processes Involved in Construction', in *Proc. of the Fourth Int. Symposium on Robotics and Artificial Intelligence in Building Construction*, Haifa, Israel, 22-25 June, pp. 322-339.

Petropoulakis L. & Malcolm C. 1990, 'Programming Autonomous Assembly Agents: Functionality and Robustness', 7th ISARC, Bristol, pp 431-438.

Rosenbrock, H. 1990, *Machines with a purpose*, Oxford University Press, Oxford.

Rosheim, M.E. 1994, *Robot Evolution-The Development of Anthrobotics*, Wiley & Sons, NY.

Selvestrel, M. C. & Goss, S., 'Optimal Flight Paths for Aircraft: A Little Knowledge Goes a Long Way', in *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Proc. Eighth Int.Conf.* Melbourne, Australia, 6-8 June, pp. 687-695.

Trevelyan, J. P. 1992, *Robots for Shearing Sheep - Shear Magic*, Oxford University Press.

Winstanley, G. 1991, *Artificial Intelligence in Engineering*, Wiley & Sons, Chichester, England.

Winter R.L., O'Brien J. B. & Wakefield R. R. 1995, 'A New Schema for the Engineering Design of Autonomous Agents in Non-manufacturing Domains', in *Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Proceedings of the Eighth International Conference*, Melbourne, Australia, 6-8 June, pp. 21-26.

Zadeh, L. A. 1996, 'The Role of Fuzzy Logic and Soft Computing in the Conception, Design and Deployment of Intelligent Systems', a seminar presented at the University of New South Wales, Sydney, Australia, 17 January.