Scheduling for and integration of an intelligent robotic handling system

Z. Katz,[1] **M. Katz**[2] **and D.C. Levy**[3]

(First received November 1994; Final version January 1995)

Abstract

An automated manufacturing system using industrial robots, possesses an enhanced flexibility and efficiency as a result of the use of intelligent materials handling systems. A smart rotating platform, incorporated under hierarchical computer control is described. The hardware development leading to the implementation of a suitable algorithm is presented for a case attempting an exact solution aimed at optimizing scheduling and active time. The use of a branch-and-bound method, preceded by a simple heuristic to determine the upper bound, is discussed. The operation of a smart materials handling system, its support hardware and its interfacing with an assembly robot as administered by a PC-AT is detailed.

Introduction

A robotic system for components handling, as applied in an automated assembly environment, requires a variety of subsystems for flexible and efficient performance. It may incorporate aspects of intelligent operation, optimized scheduling and use of expert systems. At times, when the components manipulated are of a non-standard nature, the difficulties are evident particular with non-uniformity in shape, dimension, accuracy and operational features. The cost becomes prohibitive and hence a decline in flexibility.

A manufacturing environment requires, in general, a responsiveness to unforseen or fast changing circumstances. Computer-aided process planning is one approach to deal mainly with problems related to specific requirements connected to planning of and scheduling for process performance [1] or selection of manufacturing alternatives while generating a non-linear process plan and simulation systems for process scheduling [2; 3]. Concepts of optimal assembly planning via genetic algorithms [4] contribute to the analytical approach in optimization of processes.

For specific processes such as robotic assembly, the scheduling of components delivery, handling and manipulating could contribute to an enhanced productivity and extended flexibility. A scheduling policy must respond to discrete variations, from equipment failure to demand changes, whilst possessing a computing capability easy for implementation in real-time. A dynamic simulation model for scheduling planning coupled with the development of an experimental methodology for simulation validation, offers a practical, low-cost approach for problems related to components delivery scheduling in robotic assembly applications. This work presents an attempt in this direction.

The intelligent handling approach

There are many materials handling systems available today. Most, however, are of a complex and costly nature putting them beyond the reach of the smaller volume producer with limited financial resources. This paper presents an investigation of the conceptual design of a simplistic handling approach which sacrifices costly hardware overheads while maintaining inherent intelligence under software control.

The problem tackled deals with the accurate presentation of selected components to one or more assembly robots in an optimal schedule, minimising total delivery platform rotation time. The materials handling platform chosen for investigation is of a small scaled model of a carousel-type platform, divided into six designated station zones. The platform is rotated by a stepper motor driven by supporting circuitry and directed by a PC-AT controller which also provides the intelligent decision-making necessary for scheduling optimality. A certain product has to be assembled by an industrial robot. Space is limited on the shop floor and therefore, the conveyor system needs to be of a small size. The product to be assembled may be one of several small parts, with each possible part comprised of a unique sequence of individual components placed on and presented by the platform. The part selection for assembly is left to the discretion of the system operator.

Such a system would require an elaborate feeding subsystem in order to provide the rotating platform with an accurate supply of suitable components. These feeders would require industrial standard and costly sensors. Simulation by software, of both the feeding to and the removal of components from the platform, allows for a greater freedom in design. This simulation enables greater flexibility in the system design and also permits a wider choice of suitable components. This simulation is made as realistic as possible by modelling the pseudo-random feeding techniques of existing bin-feeding technologies in order to present accurate representations of such devices. Moreover, if such simulation methods are not used, the resulting degradation of sensing capabilities would restrict both the number and type of components usable for assembly of any part. The sensing resolution would be too low and consequently, component differentiation would have been

[1]Professor, Rand Afrikaans University, Johannesburg (Member)

[2]Post-graduate student, University of Natal

[3]Senior Lecturer, University of Sydney, Australia

the number and type of components usable for assembly of any part. The sensing resolution would be too low and consequently, component differentiation would have been severely hampered. Moreover, the size, shape and orientation of the components on the platform could be highly restricted. All these factors justify the choice of simulation as an alternative to the real-time feeding and picking from the test platform.

The experimental hardware setup

The heart of the system lies with the platform itself. The platform comprises a small diameter disk built as a belt-driven platform on top of a rotating shaft. The toothed belt prevents excessive slipping and backlash and the mechanical assembly is designed to minimise rotational friction between the moving parts.

The number of designated zone areas on the platform was set as six. A larger number would have made the system impractical because the zone surface area would have been too small to facilitate carrying a sufficiently large component. The platform surface is divided into six equal segments. The robotic cell comprised three major components: the AT controller, the handling platform and the 'simulated' robot arm and bin-feeder. Interfacing between these three major components has to ensure a steady and accurate data flow. The AT controller is linked via cable to the platform motor-drive interface circuitry, enabling it to direct the platform movements under software control.

Figure 1 refers to the flowchart of hardware and signal flow in the experimental system setup. The three main hardware components comprise the AT controller, the rotating platform and its supporting hardware, as well as the robot arm for picking.

The user is prompted for part selection following which the suitable components required for the designated part assembly are fed onto the rotating platform stations.

If, for example, the user selects part **A** for assembly, and its unique four components sequence is defined as **x**, **y**, **z** and **x** (where x, y and z are from a larger set of unique possible components), then pseudo-random placements of only these types of components will occur on the platform, neglecting those other unique components not required for part **A**'s completion. As the simulation of bin-feeding ensures a pseudo-random component feed to the platform the final assembly objective is determined as either feasible or not, given the real-time component distribution on the platform. The BAB-algorithm-based software determines either a total optimal schedule in the case of all the required components being present on the stations, or else a partial schedule in the case of an incomplete set of components being present. At the completion of such a partial optimal schedule, renewed feeding is prompted and scheduling is continued, until the final component required for part **A**'s assembly is picked off by the robot arm.

Although the experimental system did not include a physical bin-feeder and robot arm the software simulation of these devices provided an embedded virtual flow of data between these system components. Control signals for the stepper motor was generated by software and implemented via a PC-30 interface card over a serial link. The PC-30 card allows for 48 input/output lines by incorporating two 8255 PIO chips on board. The drive electronics, however, required only two signal lines, one each for the platform-rotation's direction and duration.

The algorithm

Several options for solving the problem of optimal platform time utilisation were considered. Such issues included the analysis of the cost function complexity, the complexity of the search space and the practicalities involved.

The cost function is taken as the number of rotation steps, which would involve a simple summation process that could be completed in 0(n) time, meaning a number of steps proportional to the problem size. The search space, (the universal set of options that have to be inspected for the optimal) was of an exponential complexity (including factorials). The analysis requires further refinement by specifying whether the components would be replaced or not after picking.

The problem was assumed to be NP-Hard because this is the usual case for problems with easily calculable cost functions and exponential search spaces.[5] In the case of NP-Hardness, the further selection of algorithms depended on such factors as the size of the problem (how many components are present on the platform), the speed at which optimisation had to be achieved, the equipment available (an embedded solution such as an 8051 or a PC such as an 80386) as well as the quality of the solution required, in other words, was a schedule within a few percent of optimal sufficient? The problem size is small enough in nature to be able to achieve an exact optimal solution. (Six components of four unique types assured a small enough problem size.) For such a solution, a branch-and-bound type algorithm involving relatively few lines of code while using recursion is selected. The branch-and-bound method (BAB) used implicit enumeration methods (or a tree search) which can find an optimal solution by examining the subsets of feasible solutions.

Several precautions have to be taken prior to implementation of such an algorithm. The efficiency of the BAB algorithm is determined by the method of selection criteria in the branching of the subsets, successively starting from all feasible solutions. The more precise the lower bound of the subset was made, the fewer the number of nodes that had to be searched. Since the total computation time became large with complicated lower bounds, there was a distance tradeoff between preciseness and the complexity of the lower bound. Consequently, a simple heuristic was included to precede the BAB algorithm in order to determine an upper bound, as this would speed up the total computation time. A greedy heuristic seems to be the simplest.[6]

The required pick-up sequence is listed. Accordingly, components are picked up from the nearest position on the

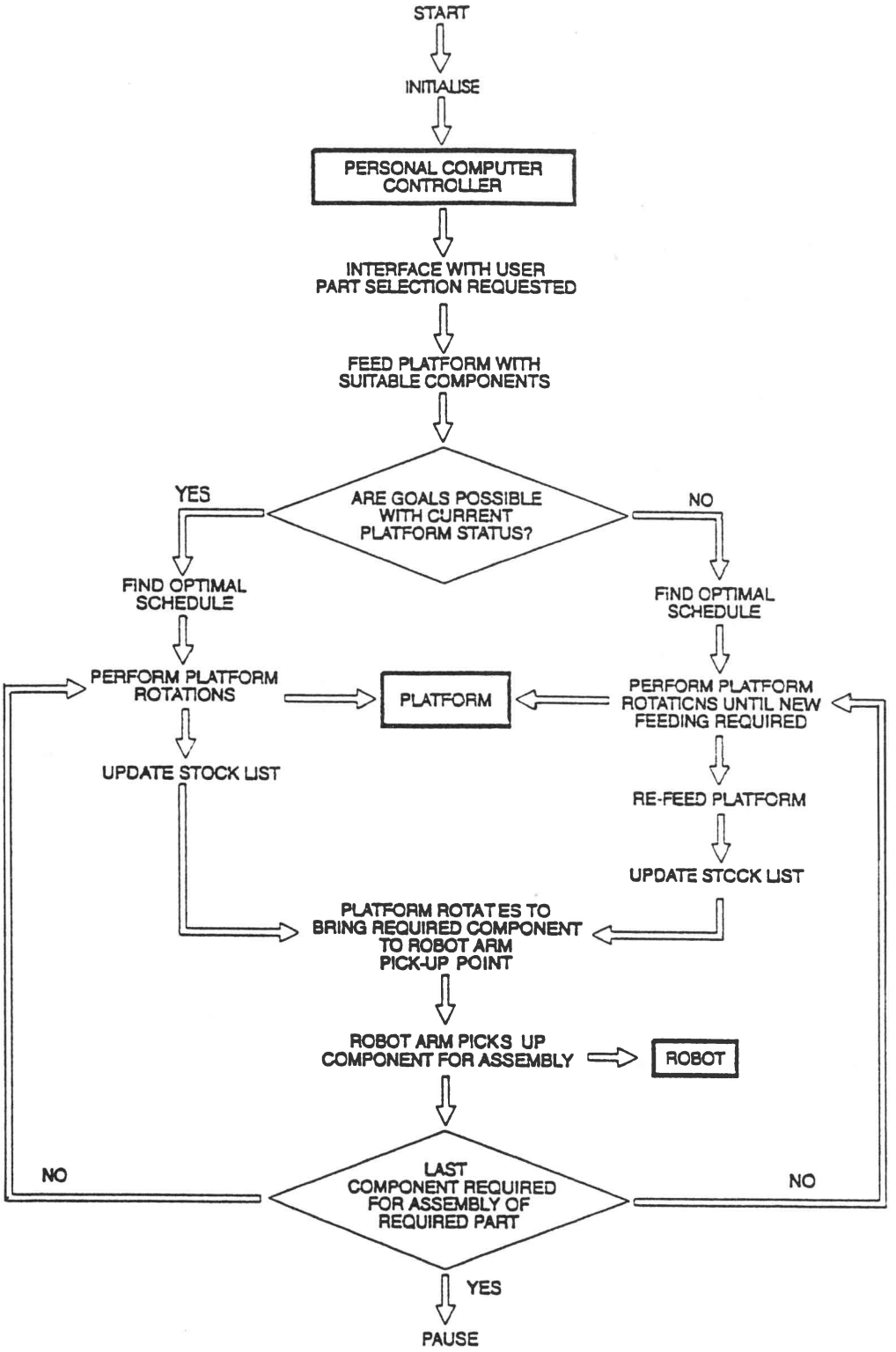

Figure 1 Hardware and signal flow

platfom recording the number of platform rotations as well as the order in which this number of rotations is achieved. The solution, although not necessarily the best, is not the worst case either.

This solution is called MAX. Then, all possibilities are enumerated, with calculations for the number of rotations for both full as well as partial choices. If any partial choice is found to have rotations that are larger than MAX, enumeration from that partial choice onwards ceased, automatically eliminating all choices that would have followed. This is termed pruning, which discards futile routing of searching space. If one particular full choice required less rotations than MAX, MAX is set to this new value and the process renewed until all possibilities were either investigated or eliminated by pruning.

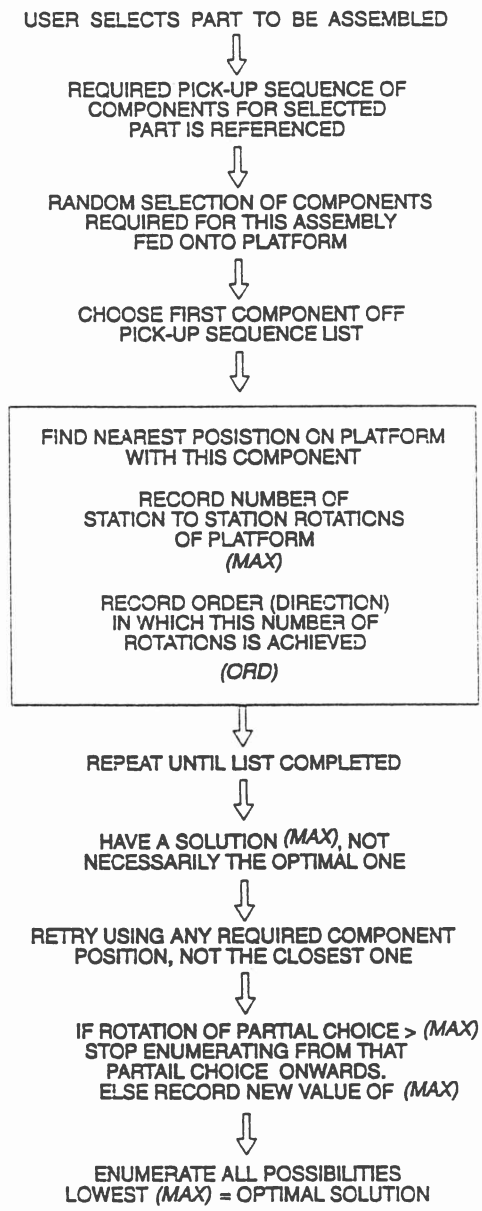

Figure 2 Logic of decision making process for part assembly

Figure 2 shows the logic of the decision-making process for a part's assembly to be successful. It is self-explanatory, although the choice of variables MAX and ORD provide a valuable clue which leads to the implementation of a branch-and-bound technique. MAX will contain the total number of platform rotations in the case of an optimal scheduling solution, and ORD will contain a record of directional information showing the sequence of clockwise and anti-clockwise rotations. Whereas MAX is the final optimal and therefore objective solution, ORD is included for testing and statistical analysis purposes and bears no direct indication on the overall system performance.

It became evident that the platform size was not an input to the algorithm implementation. Recursion in the enumeration is not needed and a number of nested loops equal to the number of components in the part being assembled are implemented instead. Each loop enumerated the possible selections from the platform and pruning is achieved by the manipulation of the index in the 'for' loops within the routine.

An analytical approach to the *BAB* algorithm

If f is the objective which requires minimization, then the Branch-and-Bound method excludes the subsets found not to include any optimal solution; and then leads to at least one optimal solution. The priority of the corresponding node is decided by the lower bound (LB) of the values of f of feasible solutions contained in each subset. Usually, a node with the least LB is branched further to decompose the related subset into its own further subsets.

An upper bound of the minimum f_0 of the objective function f is defined as the minimum f^* of the values of f of all feasible solutions found up to date. If no feasible solution is known, then the initial f^* is taken to be infinity, until the acquisition of the first feasible solution.

After the search ending at a subset which also gives a feasible solution, the algorithm needs to backtrack to the nearest incompletely searched node with an LB less than the present f^*. This node is again branched to new nodes (subsets). Hence, exclusion of subsets with no optimal solution is made by excluding any node (subset) with LB not less than the upper bound, f^*. Backtracking in the solution tree causes the branching of an incompletely searched node only in case there exists a node with LB less than the then f^*, and if LBs of the newly created nodes are greater than or equal to the then f^*. Then, backtracking is again used to search further incomplete nodes.

Finally, when lower bounds of all the incompletely searched nodes in the solution tree are greater than or equal to the then f^*, a feasible solution with this f^* is an optimal solution.

It can be proved that the BAB method determines an optimal solution after the search of a finite number of subsets (nodes). The number of nodes created in the solution tree is finite, since the original problem had a finite number of feasible solutions and each node in the solution tree is branched to at most, a finite number of nodes. Then, by branching under the condition LB $< f^*$ finally lower bounds at all incompletely searched nodes become greater

than or equal to the then f^*. All lower bounds of the nodes branched from such incompletely searched nodes are also greater than or equal to that f^*, by the fact that,

$$LB\,[Sj] - LB\,[S_i] \qquad (1)$$

where $\mathbf{S}_j - \mathbf{S}_i$ for any two subsets S_i and S_j of all feasible solution sets. (1) can be proved by noting that S_j contains a part of the set of feasible solutions. Hence, a feasible solution with that f^* becomes an optimal solution with f_0, that is,

$$f* = f_0$$

after the search of a finite number of nodes.

Optimization of scheduling

In this particular design problem it is noted that there are a finite number of distinct processing sequences, with each sequence defining a unique schedule. Moreover, it can be shown and proven mathematically that an optimal scheduling solution will always exist as a function is minimised over a finite set.[7] In this particular case study, there is no need to consider typical scheduling data that are usually of paramount importance, namely processing time, ready times and due dates, as this is a conceptual and simplified case under investigation. The conceptual approach made simple yet effective assumptions which negated the need for further investigations into these terms. It is important to distinguish between information that is known in advance and that which is generated as the result of the scheduler's decisions, such as the optimal schedule itself

The absolute need for optimization of platform scheduling is a direct result of the selection of an intelligent system approach over, say, a dedicated approach. The software alone determines the movement of the platform and its components, and the selection emphasis is removed from the duties of the feeders and pickers. The simulated feeder provides a pseudo-random supply of components to the platform from those that are required for the selected part's assembly.

The obvious advantage of having such a software-driven handling system lies with its inherent modification capability. The parts to be assembled can be varied and modified with no need of any hardware adjustments. Rather, the shop-floor operator instigates changes via the software interface allowing for quick and user-friendly modifications with no overhead. The optimization of scheduling which was currently investigated for total time/rotation minimisation can be redirected, with little alteration to the software, to include other scheduling criteria that have been omitted in this approach because of its conceptual nature alone.

Dynamic programming is an optimization technique related to a multistage decision problem. It is an implicit enumeration approach (similar to BAB) and can be very useful in reducing the computational effort. However, its large number of intermediate calculations that have to be recorded, reduces its attractiveness for problems of optimization arising from scheduling requirements.

Dijkstra's algorithm, as an alternative method for optimization, is appropriate for the finding of shortest paths particularly in cases similar to weighted directed graphs. The optimization problems, relating to roads, communication or telephone networks cases using this method, are far from similar to our case of optimal parts delivery for their predetermined assembly procedure.

Heuristic methods of optimization were also considered and sub-optimal solutions produced by such a heuristic could play an important role as an initial upper-bound solution. The BAB technique uses this as a complementary step.

The use of BAB as the optimization technique in this work was based on the fact that being a relatively small-size problem, the optimization of parts scheduling for flexible assembly is performed via an attempt for an exact solution.

Discussion

The system design is a conceptual one and serves as initial research into what is a fully upgradable intelligent robotic handling system. The results prove the initial concepts to be both valid and practical. However, several obstacles faced in this design approach need to be addressed. These include problems arising from a more complex variation in possible component orientations and dimensions. The initial approach assumes an ideal case of perfect orientation in the placing of all components on the platform by the feeding mechanism, achieved by software simulation. In a more practical approach, actual bin-feeding inaccuracies need to be overcome. Any reject component on the platform will need to be removed from its station and by doing so could waste valuable assembly time.

The system's main advantage lies in its inherent capability of accommodating a wide variety of assembly technologies. The exact feeding strategy is interchangeable with no major software adjustment required. Knowledge-based or expert systems can be incorporated into the overall system performance and to ensure a gradual learning process. The small scale design approach, with a correspondingly small scheduling solution-tree size, will shorten learning time in such an expert system and enhance overall optimal scheduling execution time.

The software features can be modified in order to vary factors for minimization. In this prototype system, the dominant factor is taken to be the 'total platform rotation steps'. This could be altered to allow for other constraints to be minimized. It can be achieved on-line, providing dynamic optimization of system scheduling. The modular design ensures that any failures are isolated to minimize their effects on the remainder of the system. Moreover, such modules can be used as building blocks in the construction of larger cells or systems.

The section of the research presented in this paper displays its initial stages. Additional work is performed to-

wards the reduction of simulated steps, introduction of advanced hardware, pattern recognition software, improved assembly cell management, artificial intelligence methods, expert and learning systems and use of approximation techniques for scheduling of large numbers of assembled parts.

Acknowledgement

The authors wish to acknowledge the Foundation for Reseach Development support via its Manufacturing Technology research grant.

References

[1] Cryssolouris G & Chan S. An integrated approach to process planning and scheduling. *CIRP Annals*, 1985, **34**, 1.

[2] Kruth JP & Detand J. A CAPP system for nonlinear process plans. *CIRP Annals*, 1992, **41**, 1.

[3] Larsen NE & Alting L. Requirements to scheduling simulation systems. Proceedings of Summer Computer Simulation Conference, Canada, 1990.

[4] Wong H & Leu MC. Adaptive genetic algorithms for optimal PCB assembly planning. *CIRP Annals*, 1993, **42**, 1.

[5] Talarage J & Hannam RG. *Flexible manufacturing systems in practice, applications, design and simulation.* Marcel Dekker, 1988.

[6] Caffman EG. *Computer and Job/Shop scheduling theory.* John Wiley, 1976.

[7] Baker KR. *Introduction to sequencing and scheduling.* John Wiley, 1974.