this, it is conceivable that it could be done automatically by using the Prolog system. This, in effect, points to the prospects for systems that can learn. Future work will seek to develop this aspect.

ACKNOWLEDGMENT

We gratefully acknowledge a generous grant from the International Development Research Center, Ottawa, Canada, which enabled this work.

REFERENCES

(1) Faulkner, L. R.; Eklund, J. A. *Proceedings of the Electroanalytical Symposium*; BAS: Chicago, 1985; p 225.
(2) Nau, D. S. "Expert Computer Systems". *Computer* **1983**, *15*, 63-84.
(3) Gunasingham, H. "Heuristic Approaches to the Design of a Cybernetic Electroanalytical Instrument". *J. Chem. Inf. Comput. Sci.* **1986**, *26*, 130-134.
(4) Kowalski, R. "Logic Programming". In *Information Processing 83*; Mason, R. E. A., Ed.; Elsevier: New York, 1983.
(5) Fuchi, K. "The Direction the FGCS Project Will Take". *N. Gener. Comput.* **1983**, *1*, 3-9.
(6) *Turbo Prolog Owner's Handbook*; Borland International: CA, 1986.
(7) Clocksin, W. F.; Mellish, C. S. *Programming in Prolog*; Springer-Verlag: Berlin, 1981.
(8) Georgeff, M. P. "Procedural Control in Production Systems". *Artif. Intelligence* **1982**, *18*, 175-201.
(9) Georgeff, M. P.; Bonollou, U. "Procedural Expert Systems"; *Proceedings of the 8th International Conference on Artificial Intelligence*; Karlsruhe, FRG, 1983; pp 151-157.
(10) Barr, A.; Fegenbaum, E. *Handbook of Artificial Intelligence*; William Kaufman: Los Altos, CA, 1982; Vol. 1.

TORTS: An Expert System for Temporal Optimization of Robotic Procedures

T. L. ISENHOUR

Department of Chemistry, Kansas State University, Manhattan, Kansas 66506

P. B. HARRINGTON*

Department of Chemistry and Geochemistry, Colorado School of Mines, Golden, Colorado 80401

Received September 9, 1987

The Temporal Optimizer of Robotic Task Sequences (TORTS) expert system has been devised as a programming aid and a precursor to the merging of laboratory robotics and artificial intelligence. This program predicts the feasibility and run times of various robotic program configurations rapidly, without requiring the robotic system. The TORTS system can evaluate different sequences of robotic tasks seeking to minimize the procedure completion time and efficiently allocate resources.

INTRODUCTION

Most laboratory procedures may be decomposed into a series of tasks. Each task requires certain resources, reagents, equipment, instruments, or personnel. The order in which the tasks are completed is important, because it will effect the overall completion time of the procedure. Scheduling, queuing, and network theories have been devised to maximize the efficiency of one or more procedures by sequencing tasks and allocating resources.[1-3]

Programming robotic systems to run laboratory procedures efficiently is often both tedious and time-consuming. Furthermore, once an operating robot program is obtained, the programmer is often reluctant to modify it. Variable laboratory procedures alter their flow on the basis of tests acquired during run time. For variable or one-time procedures, the time required to program laboratory robotic systems may exceed the time required for the chemist to do the procedure manually. For this reason current robot applications in the laboratory are usually limited to nonvarying procedures repetitiously applied to a large number of samples.

Most laboratory procedures are often variable and are applied to a limited number of samples. Variable robotic procedures would be feasible if efficient robot programs could be rapidly developed. The Temporal Optimizer of Robotic Task Sequences (TORTS) expert system has been developed as a programming aid for versatile and facile robot program development.[4,5] An advantage of the TORTS system is that a programmer can rapidly evaluate a robotic program configuration on a computer without requiring the robotic system. For variable robotic programs each procedure may be optimized separately.

The TORTS system is an expert system devised to maximize the productivity of laboratory procedures, shorten robotic program development times, and acquire knowledge of robotic systems. This system is an integral step in the development of the Analytical Director project.[6]

Chart I

```
REM
REM*** GET A TUBE
REM
  RACK.INDEX=1
  GOSUB GETTUBE.SUB
REM
REM*** FILL TUBE WITH REAGENT
REM
  SYR.N="C"
  SOL.VOL=3.0
  GOSUB ADDSOL.SUB
REM
REM*** MIX SOLUTION
REM
  MIXTIME=120
  GOSUB MIX.SUB
REM
```

ROBOTIC SYSTEM PARADIGM

The optimization of complex robotic procedures is a formidable problem. The TORTS system allows the programmer to simulate the operation of a laboratory robotic system rapidly on a computer. The TORTS system will generate robot task schedules. A schedule contains the predicted starting and completion times for each task in a procedure. Many tasks schedules may be evaluated because computer simulation is much faster than actually running and timing robotic procedures. The best task schedule may be defined as the one producing the shortest project completion time. However, cases may arise where other constraints must be met at the

0095-2338/88/1628-0215$01.50/0

Chart II

```
REM MIX.SUB
REM
REM*** ROUTINE TO PLACE A TUBE IN VORTEX, MIX
  AND REMOVE TUBE
REM
  IF FREE(MIXTIME) THEN MIXTIME = 15
REM
REM*** PLACE TUBE IN VORTEX
REM
  CLEAR.VORTEX
  OVER.VORTEX
  IN.VORTEX
REM
REM*** RELEASE THE TUBE
REM
  GRIP=120
REM
  VORTEX.ON
  SLEEP(MIXTIME)
  VORTEX.OFF
REM
REM*** REMOVE TUBE FROM VORTEX
REM
  GRIP=35
  OVER.VORTEX
  CLEAR.VORTEX
```

cost of longer completion times.

Tasks are defined as a set of robot commands that are best executed sequentially without the insertion of other commands. The definition of a task is somewhat arbitrary, and the division of a procedure into tasks depends largely on the discretion of the programmer.

A useful definition of a task is the generalized subroutine. A robot may be easily programmed to handle diverse laboratory procedures if a generalized set of subroutines is built. Chart I contains a section of a robotic program written in the Analytical Robotic Telecommunications Software (ARTS) robot control language.[7] Chart II is an example of a generalized subroutine to mix a solution in a tube for a specified amount of time. The generalized subroutines are customized for a specific need by setting global variables in the main program. The global variable, mixtime, in Charts I and II specifies the time that the vortex mixer is to run. Any complex laboratory robotic procedure may be composed of such generalized subroutines.

A task may be composed of several subroutines if the sequence of subroutines is uniquely defined. Combining subroutines into tasks will simplify the output and increase the search rate of the solution space. The search will not consider potentially optimal solutions that require reordering the subroutines contained in a task.

If a robot control module could direct and queue commands to more than one module, then tasks could execute simultaneously. Most laboratory robotic control systems either allow only one module to be controlled at a time or do not allow commands to be sent and queued at each module. Instead, the programmer is required to manually merge robot commands to execute tasks concurrently.

The TORTS system has a switch that allows it to operate in single- or multiple-task modes. The single-task mode is compatible with existing robot control technologies and allows only one task to be executed at a time. This feature eases the programming of the robot at the cost of longer run times because tasks do not have to be merged. In the future, laboratory robot control systems should provide a routing and queuing system so that packets of robot commands (tasks) may be sent to modules allowing concurrent task execution.

The TORTS system uses discrete time increments. The system will only examine times when the robotic system changes states. The algorithm advances in steps, where each step corresponds to a task being initiated. The steps will be referred to as the level of the search. The TORTS system must evaluate and store all the feasible tasks at each level. The TORTS system must also consider additional tasks that may be initiated while the robotic system waits for other tasks that are executing to be completed.

Table I. Orientation Time Matrix (Seconds) for Primitive Tasks of the Enzyme Assay Robotic Procedure

	1	2	9	10	11	12	13
1	1.2	1.2	1.2	1.2	1.2	1.2	1.2
2	1.2	1.2	1.2	1.2	1.2	1.2	1.2
9	1.2	1.2	11.0	47.6	49.0	1.2	49.0
10	1.2	1.2	47.0	14.0	49.0	1.2	15.0
11	1.2	1.2	8.0	45.0	11.0	1.2	49.0
12	1.2	1.2	1.2	1.2	1.2	1.2	1.2
13	1.2	1.2	43.0	7.8	45.0	1.2	12.0

The TORTS system uses two classes of time intervals. The first time interval is the task time, which is the duration of a task. The task time does not depend on the sequencing of the tasks. The task time will vary as a generalized subroutine is customized for a specific laboratory procedure.

The second time interval is orientation time, which is the time required for the orientation of the robot system from a present state, some task, to a future state, which is the starting conditions of the next task. The orientation time depends on the sequence of tasks. The orientation time contains information regarding the locations of modules on the robot table. Therefore, the orientation times will be constant for all laboratory procedures as long as the layout of the robot table is not changed.

The orientation times are stored as a time matrix, where the rows represent the current state of the robotic system and the columns represent the initial state required by a task. The element $t_{i,j}$ represents the time required to go from the current state i to position j. The orientation times only have to be defined in terms of primitive identification (PI). Primitive identification identifies modules by their spatial position on the robot table. Table I is an example of a time matrix. Note all the diagonal elements are greater than zero. The time required to send a command from the computer to the robot control module is 1.2 s. These times correspond to tasks that do not depend on the orientation of the robotic system. The larger diagonal times are for tasks that start and end with different orientations of the robotic system. For example, solvent delivery systems may have to refill after use.

The orientation time matrix may be built by measuring the time required to move from the end state to the start state of all the tasks that will be used. Once a time matrix has been obtained, it will not change unless the locations of the modules are moved, the robot gripper is sped up, or the robotic system is reconfigured in some other manner.

The task times will vary for different applications and may have to be obtained each time the generalized subroutines are customized. Commands may be built into the generalized subroutines that will store their tasks times for future use. The robot programmer may estimate the task times if time constraints are unnecessary. The task times are independent of sequencing and an optimum sequence should be obtainable, although the predicted times will depend upon the accuracy of the programmer's estimates.

Certain constraints exist for sequencing tasks. The tasks must be sequenced in a logical order. In order to mix solutions in a tube, the robot gripper must get a tube, add the solutions to the tube, and then mix the solutions in the tube. If any of these three tasks are interchanged in the above sequence, the procedure will fail.

Tasks have a limited capacity. Examples of capacitative constraints are a centrifuge that holds six tubes, a vortex mixer

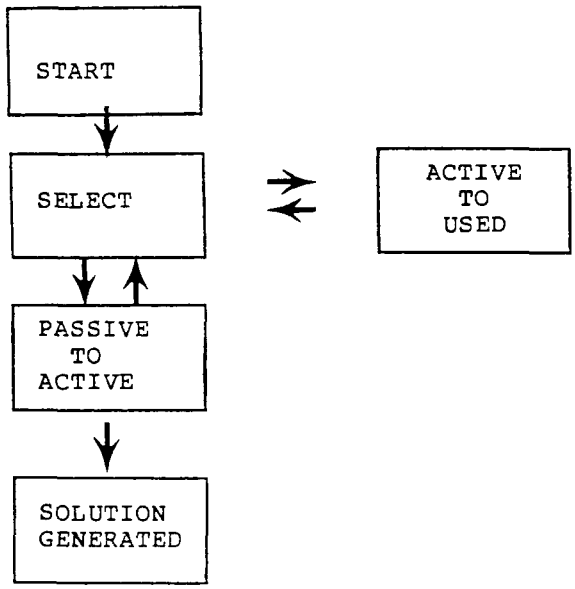

Figure 1. Flow chart for the solution generator of the TORTS program. SELECT directs the algorithm to the two modules PASSIVE TO ACTIVE and ACTIVE TO USED. A solution is generated when the PASSIVE and ACTIVE lists are empty.

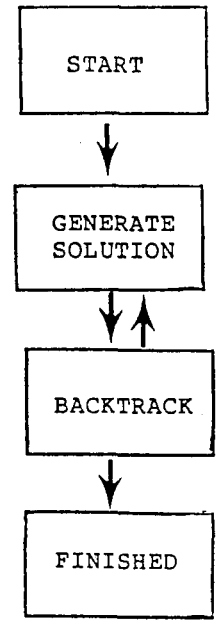

Figure 2. Flow chart for the exploration of the solution space. GENERATE SOLUTION and BACKTRACK initiate each other alternately until all the stored tasks are explored at each level and the system terminates.

that holds one tube, and a robot gripper that grips one tube. The PI is also used for maintaining the capacity of modules during the robotic system simulation.

Procedures may have timing constraints for tasks, including required time intervals between tasks and tasks that must be completed by certain times. Time constraints are crucial for kinetic studies.

A lock time is the time penalty for interrupting other executing tasks using the same module. After interrupting a task with a lock time, that task must be reinitiated. If the centrifuge is running, an amount of time is required to stop it, add another tube, and restart it.

The TORTS system may be divided into two parts, solution generation and solution-space search. The solution generator generates feasible solutions and stores the elapsed time in the same manner as an accountant. A feasible solution is a sequence of tasks that violates no constraints. The solution generator uses three primary lists, the passive list (PASSIVE), the active list (ACTIVE), and the used list (USED). The PASSIVE list contains tasks that are unconstrained and that are not running. The ACTIVE list contains tasks that are running. The USED list stores tasks that have been completed.

Figure 1 is a flow chart of the TORTS solution generation algorithm. The TORTS algorithm consists of several modules. A solution is generated by the following procedure. The module START fills the passive list with tasks that are not logically constrained. The SELECT module evaluates all the tasks on the passive list and selects the "best" task to be initiated at this level.

Slack time is defined in network theory as the maximum time a task may be delayed without increasing the procedure completion time and is calculated for a task by subtracting the earliest time from the latest time. Another time interval, wait time, is the time the robotic system must wait so that it will not violate any timing constraints. The slack times are obtained from a critical path analysis (CPA).[8] The slack time is added to the orientation time plus any time penalties from constraints, lock times, or wait times. All these times are weighted equally because the same time units are used.

The best task is defined by the shortest time required to initiate the task plus its slack time. This task is sent to PASSIVE to ACTIVE, which moves tasks from the PASSIVE list to the ACTIVE list. The time is updated by the orientation time required to initiate the task. The algorithm returns to SELECT, and if not other tasks are feasible, it will go to ACTIVE TO USED, where completed tasks are moved from the ACTIVE list to the USED list. If the completed task has any successors, they may become logically feasible and will be added to the PASSIVE list. The algorithm will return to SELECT. This process continues until the PASSIVE and ACTIVE lists are empty. A feasible solution is generated, and the times for each task are obtained.

OPTIMIZATION

The TORTS system uses a depth-first search of the space of feasible solutions. The depth-first search is directed by heuristics and expert strategies.[9,10] Figure 2 outlines the optimization procedure. Solutions are generated one at a time. Each improved solution lowers the upper bound on the predicted completion time. Each time an improved solution is found or the upper bound is exceeded, the system will backtrack and try an alternative feasible task. The optimization is completed when the system backtracks to the top and all feasible task sequences have been evaluated.

A heuristic is a rule of thumb or a relaxation of constraints to simplify a problem. For the purposes of this work, complex heuristics based on task schedules obtained by relaxing one or more constraints will be referred to as expert strategies. Figure 3 shows the hierarchical search strategy that the TORTS system uses. The first search is a critical path analysis. Next an optimum task sequence is obtained for a one sample or duty cycle procedure. The last search optimizes the multiple sample or duty cycle procedure. The advantage of the hierarchical search is that each search generates new heuristics for controlling searches of the larger solution spaces.

The search space is represented as a tree. The level is the depth of the tree, and the TORTS system systematically backtracks to search the entire solution space. The nodes in the tree consist of tasks that violate no constraints. The solution generator stores all the alternative tasks in the tree during its depth-first search of the space. The alternative tasks are tasks that are feasible or will become feasible if the robotic system waits for tasks on the ACTIVE list to be completed. The times the robotic system must wait are also stored at each level.

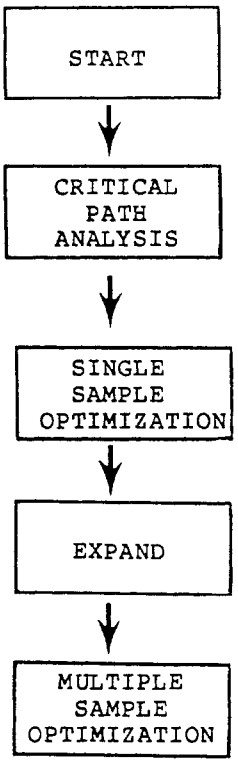

Figure 3. Hierarchical search strategy used by TORTS.

Equivalent tasks are identical operation on different samples. Pruning equivalent task sequences from the search tree reduces the time for the optimal task sequence to be found. The TORTS system uses a canonicalization function so that no equivalent tasks will be stored at each level in the search tree and equivalent task sequences explored. Equivalent tasks are found in multiple sample or complex laboratory procedures. After the first solution, the system will backtrack, exploring all the alternate tasks at each level.

If laboratory procedures do not have many constraints, the search space will increase exponentially. The search may be sped up by limiting the number of tasks stored at each level in the search tree or by pruning all tasks that are below a set priority. However, restricting the search in this manner will not guarantee the optimum solution will be found.

Evaluation functions, functions that estimate the procedure completion time, are important because they allow early pruning of nonoptimal solutions from the search. The TORTS system uses consistent evaluation functions, functions that will underestimate the procedure completion time and guarantee that optimal solutions will not be pruned from the search. Another property of consistent evaluation functions is that the estimate of the completion time increases in accuracy monotonically with the level of the search.

The first search the TORTS system uses is a CPA to obtain task slack times and evaluation functions. CPA is an example of an expert strategy. All the capacitative constraints are relaxed. Therefore, all the tasks that satisfy the logical sequence of procedures may run concurrently. The CPA times are obtained from both task and orientation times.

CPA gives the earliest and latest times tasks may be initiated without increasing the procedure completion time. CPA gives an early indication if the entire procedure is not feasible because of inconsistent constraints.

Tasks are prioritized by their slack times so that tasks on the critical path will be initiated as soon as they are placed on the PASSIVE list. An evaluation function also may be obtained for each task by subtracting the CPA latest time from the CPA project completion time. The procedure completion time is estimated by adding the difference between CPA times to the current time that the task is initiated.

The entire state of the robotic system is stored at each level. BACKTRACK advances upward to levels that contain unevaluated tasks which are stored by SELECT. The algorithm proceeds directly to PASSIVE TO ACTIVE from BACKTRACK with the next best task, and that task is deleted at that level. The algorithm now advances downward to generate another feasible solution. Once an improved solution is found or the upper bound is reached, the algorithm returns to BACKTRACK. When BACKTRACK reaches a level of zero, the algorithm is finished and the solution space has been searched.

The TORTS system can expand a single-sample procedure or single duty cycle into a multiple-sample procedure or multiple duty cycles. This step of the optimization occurs after an optimum task sequence is obtained for the single-sample procedure. Optimizing multiple-sample procedures allow the duty cycles to overlap and further minimize the total completion time.

Each task of a single procedure will form a parallel task for the additional duty cycles. All the tasks will be coded by a unique integer, but parallel tasks are identified by their PI. Expanding a single procedure to multiple procedures also relaxes the constraint of the logical sequence of procedures. Consequently, the larger the number of duty cycles optimized, the greater the potential time savings.

The correct choice of the number of duty cycles to be optimized is important because the search space will increase exponentially with the number of duty cycles. The run time for the TORTS system will also increase to search the entire solution space and verify that the selected task sequence is optimum. Another consideration is the turnaround time for each procedure. If a single procedure may not wait to be completed, the optimum task sequence will be to run each procedure serially. For procedures that require many samples or duty cycles, efficient patterns of task sequences may be obtained from optimizations of a reduced number of duty cycles. The robot programmer needs to follow this pattern while programming the robot to perform the larger number of duty cycles.

Before the multiple procedure solution space is explored, the single-procedure optimum task sequence is used to prioritize the tasks. CPA is used to obtain a new evaluation function. Tasks are prioritized by the single-sample starting times. The end result is that the task sequences explored first will maximize the concurrence of parallel tasks, and task changes will follow the optimum pattern.

EXPERIMENTAL METHODS

The TORTS system is written in Digital Equipment Corp. FORTRAN 4.5 and runs under the VMS 4.1 operation system on a MicroVax II. All computations were performed by using single-precision arithmetic.

Figure 4 is a schematic of the layout of the Zymate robot system, and Chart III identifies the components of the robotic system.

An enzyme assay was chosen to evaluate the TORTS system because the timing of the spectrophotometric measurements is crucial to the analysis. The assay is the activity determination for liver alcohol dehydrogenase.[11] The method is based on spectrophotometric measurement of the amount of nicotinamide adenine dinucleotide (NAD) being reduced in 3 min at pH 9.6 in excess ethanol.

Approximately 1 mg/mL of enzyme is added to 3 mL of a 0.1 M glycine–sodium hydroxide buffer that also is 0.57 M in ethanol. A 1-mL aliquot of a 1 mg/mL solution of NAD is mixed with the enzyme buffer solution, and the absorbance at 340 nm is promptly acquired. A second measurement is

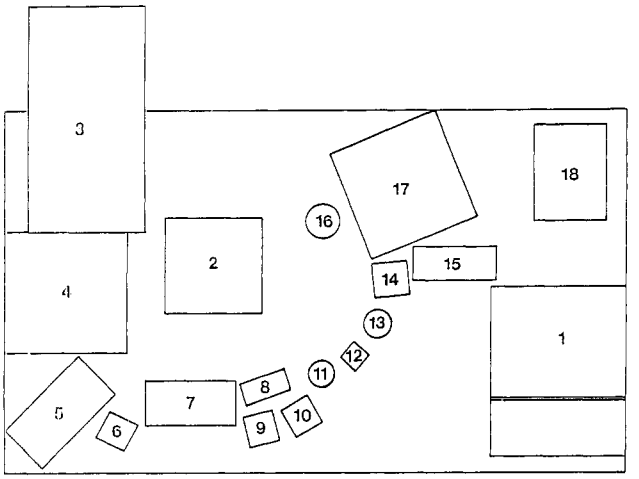

Figure 4. Schematic of the robotic system. Chart III identifies each module.

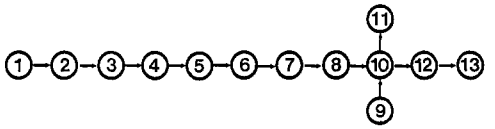

Figure 5. Unidirected graph of the logical flow of the enzyme assay tasks.

Chart III

(1) Zymate laboratory controller equipped with a Z-840 computer interface
(2) Zymate laboratory robot
(3) Hewlett-Packard 8451A diode array spectrophotometer
(4) S/P water bath
(5) Mettler AE100 balance equipped with a Z-850 balance interface
(6) Zymate general purpose hand
(7) Zymate sample rack
(8) Zymate syringe tip rack
(9) Zymate liquid distribution hand
(10) Zymate blank hand equipped with cannula
(11) contact switch station
(12) Zymate liquid dispensing station
(13) waste-dispensing station
(14) Zymate vortex station
(15) Zymate master laboratory station
(16) Zymate capping station
(17) Zymate centrifuge
(18) Zymate power and event control station

made 3 min later, and the change in absorbance is recorded. The enzyme concentration is measured at 280 nm.

This assay was decomposed into the 13 tasks listed in Table II. The average task times and their standard deviations for six replicates are also listed in Table II. Figure 5 contains a unidirected graph of the logical sequence of the tasks. Tasks 1 and 9 are initial tasks, and tasks 11 and 13 are terminal tasks. The critical path is the horizontal path from task 1 to task 13.

Tasks 9 and 11 require a general purpose gripper hand on the robot. Tasks 10 and 13 require a blank hand equipped with a cannula. The cannula is attached by a 1-mm-i.d. Teflon transfer line to a 10-mL syringe in the solvent delivery system. The cannula is necessary to fill and empty the cuvette inside the spectrophotometer.

Tasks 1–8 fill and flush the transfer line with buffer solution. Task 9 takes a tube that contains 1 mL of 1 mg/mL of liver alcohol dehydrogenase and addes the ethanol and buffer solution. The subroutine terminates by leaving the tube in the vortex mixer with the mixer on. Tasks 10 uses the cannula to transfer 1 mL of 1 mg/mL of NAD solution to the tube in the vortex mixer. After the solution is allowed to mix for 4 s, the mixer is turned off, and 2.5 mL of the NAD and enzyme solution is transferred by using the cannula to the cuvette inside the spectrometer. The first spectrophotometric measurement is made as the final step of this task.

Task 11 removes the tube from the mixer and places it back in the tube rack. Task 12 makes the second spectrophotometric measurement 3 min after task 10 is completed. Task 13 empties and flushes the cuvette inside the spectrometer with buffer solution and discards the solutions at the waste-dispensing station.

Table II

tasks	PI	times[a] (s)
(1) set syringe parameters	1	3.7 ± 0.0
(2) fill syringes	2	26.1 ± 0.0
(3) set valve to flush line	1	3.7 ± 0.1
(4) pump syringe C	2	26.1 ± 0.0
(5) set valve to fill syringe	1	3.7 ± 0.0
(6) fill syringe	2	26.1 ± 0.0
(7) set valve to flush line	1	3.7 ± 0.0
(8) flush line	2	26.1 ± 0.0
(9) get buffer, enzyme and mix	9	94.2 ± 0.9
(10) get NAD, mix and measure	10	228.5 ± 0.5
(11) remove tube from mixer	11	43.0 ± 0.3
(12) take second measurement	12	12.2 ± 0.0
(13) clean cuvette	13	228.5 ± 0.7

[a] All average times and standard deviations were calculated from six runs.

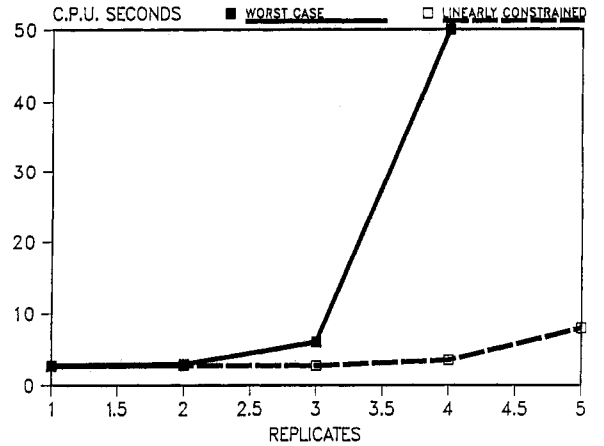

Figure 6. Graph of the CPU time (seconds) required by TORTS to optimize procedures plotted as a function of duty cycle: (■) worst case, all sequences are equivalent; (□) linearly constrained, the task sequence is uniquely defined for a single duty cycle.

RESULTS AND DISCUSSION

This type of scheduling problem requires an exponential amount of time for the worst case. Figure 6 is a plot of CPU seconds for the TORTS system to terminate as a function of batch or duty cycle. Two procedures each composed of three tasks are plotted. The first is an example of a worst case; all the task schedules are equivalent. The second case is serially constrained by the logical sequence of tasks. For the second case, only one feasible solution exists for a single duty cycle. However, as the duty cycles increase, the number of solutions increases exponentially. Changing the computing power will only effect the abscissa scale and not the exponential shape of the curve.

The TORTS system has a heuristic search option. The heuristic search does not attempt to evaluate all possible solutions to verify that the best solution obtained is the optimum solution. The breadth of the search is narrowed by limiting the number of alternative tasks at each level or by removing tasks with low priorities from the search.

The TORTS system was used to optimize a two-sample enzyme assay for liver alcohol dehydrogenase. Table II identifies the tasks in the enzyme assay and lists the task times and their standard deviations obtained over six runs. Table II contains the PI that identifies parallel tasks. The procedure

Table III. Solutions Generated by TORTS from the Single-Sample Enzyme Assay

	1		2	
level	task	time (s)	task	time (s)
1	1	0.0	1	0.0
2	2	4.9	2	4.9
3	3	32.2	3	32.2
4	4	37.1	4	37.1
5	5	64.4	5	64.4
6	6	69.3	6	69.3
7	7	96.6	7	96.6
8	8	101.5	8	101.5
9	9	128.8	9	128.8
10	10	270.4	10	270.4
11	12	624.8	11	624.8
12	13	653.2	12	623.6
13	11	926.5	13	686.0
completion times = 967.0 and 914.3				

contains tasks composed of single subroutines and complex tasks composed of multiple subroutines. The orientation times for these tasks are listed in Table I by their PI.

Table III contains the predicted solutions and start times for each task obtained from the single-sample enzyme assay. The complete search space was evaluated, and nine backtracks were completed.

The two-sample enzyme assay optimization required 792 backtracks and 16 s of CPU time. The predicted completion time for the optimized task sequence was 1793.6 s. The optimum sequence initiates tasks 11 and 24 in the 180-s wait periods between spectrophotometric measurements.

Another important advantage to computer modeling of robotic procedures is that wait times may be obtained. The spectrophotometric measurements must be separated by 180 s. The wait times are calculated from the output of the TORTS system. The wait time is obtained by subtracting the predicted completion times of inserted tasks 11 and 24 from the predicted start times of tasks 12 and 25. The predicted wait time of 88.1 s was used during the programming of the robotic system. The wait times between the completion of tasks 11 and 24 and the start of tasks 12 and 25 are needed to ensure that the second spectrophotometric measurement is completed 180 s after the first measurement acquired at the ends of tasks 10 and 23.

Table IV lists the predicted and experimental tasks for the two-sample enzyme assay robot program. The experimental robot times were averaged over six runs. The timing errors will propagate as the size and complexity of the robot program increase. The standard deviations of the predicted times increase with respect to the level or complexity of the robot program.

The experimental times between spectrophotometric measurements were 179.9 s between tasks 10 and 12 and 181.1 s between tasks 23 and 25. If the two-sample enzyme assay were run in serial mode, the completion time would be 1977.0 s. The optimized solution was 1795.6 s, providing a time improvement of 181.4 s or 10.1%. The single-sample optimized task sequence follows the same pattern as in the two-sample optimization. The heuristics that guide the search are efficient because the optimized sequence was obtained on the first exploration for the two-sample procedure.

Without using the TORTS system, the robot programmer would have to evaluate whether any tasks could fit within the three 3-min wait time between spectrophotometric measurements. In addition, the robot programmer would have to minimize the number of hand changes between tasks.

The optimal task sequence for the two-sample enzyme assay procedure is not intuitively obvious. To establish the optimum sequence, the programmer would have to foresee that the number of hand changes is reduced by running task 22 (PI = 9) from the second sample before task 13 from the first sample. Furthermore, without use of a programming aid such as TORTS, there is no way to verify that the robotic program is optimal without exhaustive testing.

Table IV. Predicted and Experimental Start Times

			times	
level	task	PI	predicted	exptl[a]
1	1	1	0.0	0.3 ± 0.0
2	2	2	4.9	5.1 ± 0.2
3	3	1	32.2	32.4 ± 0.2
4	4	2	37.1	37.5 ± 0.2
5	5	1	64.4	64.8 ± 0.2
6	6	2	69.3	69.5 ± 0.2
7	7	1	96.6	96.1 ± 0.2
8	8	2	101.5	100.6 ± 0.2
9	9	9	128.8	129.3 ± 0.2
10	10	10	270.4	271.2 ± 1.1
11	11	11	492.6	492.0 ± 1.3
12	12	12	623.6	623.1 ± 1.6
13	22	9	643.8	644.1 ± 1.8
14	13	13	786.8	787.0 ± 2.3
15	14	1	1016.3	1017.4 ± 2.3
16	15	2	1021.2	1022.0 ± 2.3
17	16	1	1048.5	1049.6 ± 2.2
18	17	2	1053.4	1054.5 ± 2.2
19	18	1	1080.7	1081.9 ± 2.3
20	19	2	1085.6	1086.5 ± 2.3
21	20	1	1112.9	1113.4 ± 2.2
22	21	2	1117.8	1117.9 ± 2.2
23	23	10	1151.7	1151.9 ± 2.2
24	24	11	1373.9	1375.2 ± 2.6
25	25	12	1504.9	1506.4 ± 2.5
26	26	13	1567.3	1565.6 ± 2.7
project completion time			1795.6	1793.9 ± 3.1

[a] The experimental results were obtained from six runs.

The TORTS system is meant to be used as a program development tool and a precursor to an automatic laboratory robot programming system. Improved robot technologies, better temporal precision, and sophisticated robot controllers that allow parallel operation of modules will increase the applicability of the TORTS system for facile and versatile robotic programming.

CONCLUSION

Laboratory robotic procedures may be efficiently programmed if the procedure is decomposed into a sequence of optimally scheduled tasks. Laboratory robotic procedures contain many constraints that are difficult for the programmer to anticipate and therefore satisfy. The TORTS system is a computer program that models the operation of a laboratory robot. The system evaluates only feasible solutions and outputs an optimal task sequence.

The task times do not have to be accurately known if timing constraints do not have to be satisfied because task times are independent of task sequence. The programmer may use estimated task times to obtain an optimized task scheduled; however, the predicted times depend on the accuracy of the estimated task times.

Additional expert strategies should be incorporated into future modifications of TORTS. These strategies will contain heuristics that would also be used by the robot programmer. Capacitative bottlenecks may be removed by using additional commands such as using the tube racks as temporary storage locations.

The TORTS system uses a discrete time model that under certain conditions may cause errors. These conditions will exist only when the TORTS and the robotic system are operated in the concurrent mode. Most laboratory robot controllers cannot be operated in this mode. Expert strategies should be

capable of detecting predicted solutions for errors created by the discrete time model and correcting them.

ACKNOWLEDGMENT

We thank Dr. J. C. Marshall and W. A. Schlieper for their valuable comments and criticisms. We thank Lenny Holmes for supplying the enzyme and NAD. This research was supported by the National Science Foundation under Grant CHE-8415295.

LITERATURE CITED

(1) French, S. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*; Wiley: New York, 1982.
(2) Cooper, R. B. *Introduction to Queuing Theory*; North-Holland: New York, 1981.
(3) Phillips, D. T. *Fundamentals of Network Analysis*; Prentice-Hall: Englewood Cliffs, NJ, 1981.
(4) Harrington, P. B.; Isenhour, T. L. "An Expert System for Temporal Optimization of Robotic Procedures". Presented at the 1986 Pittsburgh Conference, Atlantic City, March, 1986.
(5) Harrington, P. B.; Isenhour, T. L. "Expert Strategies for the Temporal Optimization of Robotic Procedures". Presented at the 1987 Pittsburgh Conference, Atlantic City, March, 1987.
(6) Isenhour, T. L. "Robotics in the Laboratory". *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 292.
(7) Schlieper, W. A.; Isenhour, T. L.; Marshall, J. C. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 159.
(8) Horowitz, E.; Sahni, S. *Fundamentals of Data Structures*; Computer Science: Taft, CT, 1976.
(9) Nilsson, N. J. *Principles of Artifical Intelligence*; Tioga: Palo Alto, CA, 1980.
(10) Pearl, J. *Heuristics*; Addison-Wesley: Reading, MA, 1984.
(11) Colowick, S. P., Kaplan, N. O., Eds. *Methods in Enzymology*; Academic: New York, 1955; p 495.

An Efficient Graph Approach to Matching Chemical Structures

O. OWOLABI

Department of Computer Science, University of Strathclyde, 26 Richmond Street, Glasgow G1 1XH, U.K.

Received February 19, 1988

A matrix minimization method for the generation of canonical orderings for the nodes of a graph is outlined, with particular emphasis on its application in matching chemical structures. By use of both the connectedness and the attributes of each node in the representative graph, the modified adjacency matrix is reduced to a canonical form. This canonical matrix is unique for each distinct compound represented. For storage and comparison, a canonical linear string is generated for each compound. All known structures are compiled only once in this efficient manner, and the string representation also permits best-match retrieval. Examples are given to illustrate the procedure.

INTRODUCTION

One of the most widespread uses of computers in chemical information systems is in the matching of chemical structures. Looking for the name and other information associated with a particular structure involves the matching of the structure against a library of known structures.

Due to the nature of chemical compounds, it is natural to map them onto the graph structure for convenience of manipulation. One of the main advantages of representing a complex structure such as a chemical compound by the graph is that the description can be in terms of the constituting primitives of the compound. In this case the primitives are the atoms present in the compounds of interest.

The matching of two graphs, however, is not a trivial problem. The most straightforward way to solve this problem is to conduct a node-to-node matching. In this scheme, all the possible mappings between the two graphs are generated. Each is then evaluated to determine whether it represents a legal mapping between the two compounds or not. This process will yield $n!$ possible mappings between two graphs with n nodes each. This is a computationally expensive procedure.

To overcome the limitation imposed by this naive approach, various alternatives have been proposed. The most obvious one is to combine the straightforward exhaustive approach with backtracking. With backtracking, a check is made every time a new node is added to a possible solution. Once a failure is encountered, that particular mapping is abandoned, and a new one is generated. In this way, all the possibilities are not exhaustively searched. This can be further improved by mapping only between nodes of the same type.

This is the basis of the node partitioning strategy employed by Unger[1] and Sussenguth.[2] To compare two graphs, the node set is partitioned into equivalence classes according to various node properties. These properties include node type, node degree, branch type, etc. Assignments are made between equivalent classes in the two compounds. New partitions are then generated. These two steps are iterated until either all the nodes have been matched or it is established that the compounds do not match.

There are $n!$ ways of ordering the nodes of a single n-node graph. Thus, the major problem in graph matching is to determine whether any of these orderings of a candidate graph are equivalent to the stored model graph. This consideration has given rise to the idea of seeking what amounts to the canonical ordering of the graphs to be matched. In this approach, only the canonical forms of the graphs being considered are compared. To check a database for an unknown structure, the search structure is reduced to its canonical form. This is then compared against the canonical forms of the stored structures to see whether there is a matching compound in storage.

Some of the earlier works in the generation of canonical orderings for graphs were carried out by Scoins[3] and Nagle.[4] The approach has been further explored by Randić.[5] The basic method is to represent an n-node graph by an $n \times n$ connectivity matrix. Each row is taken as a decimal number. The rows and columns of this matrix are then rearranged until the rows are sorted in ascending order. This is done by starting

0095-2338/88/1628-0221$01.50/0