Proceedings of the ASME 2010 International Design Engineering Technical Conferences &Computers and Information in Engineering Conference
IDETC/CIE 2010
August 15-18, 2010, Montreal, Quebec, Canada

DETC2010-28723

STATE TRANSITION IN RECONFIGURABLE SYSTEMS

Magdalena K. Chmarra
Department of BioMechanical Engineering
Faculty of Mechanical, Maritime and Materials Engineering
Delft University of Technology
Delft, 2628 CD
The Netherlands
Email: m.k.chmarra@tudelft.nl

Jacques Verriet
Embedded Systems Institute
Eindhoven, 5600 MB
The Netherlands
Email: jacques.verriet@esi.nl

René Waarsing
Océ-Technologies B.V.
Venlo, 5914 CA
The Netherlands
Email: rene.waarsing@oce.com

Tetsuo Tomiyama
Department of BioMechanical Engineering
Faculty of Mechanical, Maritime and Materials Engineering
Delft University of Technology
Delft, 2628 CD
The Netherlands
Email: t.tomiyama@tudelft.nl

ABSTRACT

Reconfigurable systems that maintain a high level of performance under changing operational conditions and requirements are an ongoing research challenge. Many existing systems are able to work under narrow operational conditions only. They should perform the configuration transitions from the current state to a desired one smoothly. The question is: how to find the optimal state transition trajectory that configures the system from an initial state to a desired state? We present a method to determine steps (list of actions to be taken) of state transitions for reconfigurable systems. This method makes use of graph search algorithms that solve shortest path problems and that are commonly used in routing: Dijkstra's algorithm and the A algorithm. The method is applied to the design of a reconfigurable printer, which has to change its configuration to achieve maximum performance (e.g. high quality of print) when operational conditions are changing (e.g. speed of printing).*

INTRODUCTION

There is a need for systems that are able to work optimally in any environment, under any state, and for any user-defined task. Many modern products are designed to be able to operate under different operational conditions (e.g. furniture, vehicles, tools) [2]-[8], their capabilities to do so are still limited. For most of these products the configuration change from the current state to a desired one often takes place when the product is not in use [1]. Many of reconfigurable and changeable (or flexible, adaptive) systems do not consider dynamic changes of their configuration [4],[10]. Presently, a considerable amount of research focuses on the design of products that can make a transition between two or more functional states [1]-[3],[9]-[16]. In addition, most of the research done in the design community focuses on changes in the mechanical configuration of systems, but not on system behavior.

The purpose of this paper is to convey our experience in developing methods to manage system behavior between states at runtime. The states are defined by system settings (e.g. variables such as temperature, velocity). It has to be recognized

Copyright © 2010 by ASME

which system input variables are adaptable and which ones are non-adaptable.

Complex systems are required to deliver high-quality work. For example, professional printers are required to deliver high-quality prints independently of the media type (e.g. paper mass, plain/glossy paper) and the job type (e.g. simplex/duplex). The print quality depends on the printing settings (e.g. temperature, pressure, velocity). Although printers are designed to perform their work optimally under different operational conditions, their "run" state is still very general and makes use of one specific setup, meaning under different conditions other than this specific setup, they run suboptimally. It is, therefore, desirable to implement reconfigurability in a printer such that it achieves the best performance in any given circumstances by widening up the allowed operational conditions.

In addition, ideally, reconfiguration (state transition) of a printer should take place at runtime. The question therefore boils down: how to find the optimal state transition trajectory that configures the system from an initial state to a desired state?

The behavior of dynamic systems is commonly handled using control theory [17],[18]. Figure 1 presents the concept of a negative feedback loop to control the dynamic behavior of a system. In this case, a sensor is used to sense an output value and subtract it from a reference value. This control scheme can be used in the professional printers for feedback loops such as temperature control and speed control. This scheme, however, cannot be applied to controlling the quality of the print, since print quality is not measured at runtime due to unavailability of special sensors. Because high-quality printing needs to be maintained in professional printers, a supervisory-level controller that pre-computes an optimal reference for the low-level controllers without feedback of the system (Fig. 2) is needed.

A schematic representation of a supervisory-level and low-level controllers is presented in Fig. 3. To pre-compute an optimal reference for the low-level controllers, it seems necessary to use predictive control. Predictive control is also known as model-based predictive control (MBPC), general predictive control (GPC), receding horizon control (RHC), sequential open-loop optimizing control (SOLO), dynamic matrix control (DMC), etc. [19]. There are three key ideas that differentiate predictive control from other control methodologies:

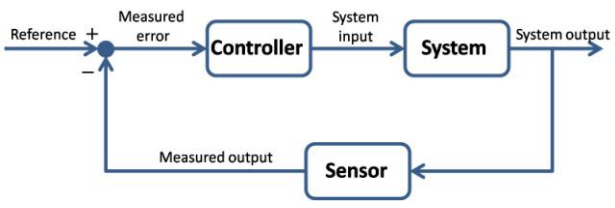

Figure 1. A TYPICAL, SINGLE-INPUT, SINGLE-OUTPUT FEEDBACK LOOP WITH DESCRIPTIONS FOR ITS VARIOUS PARTS

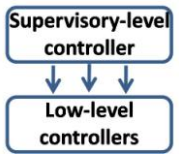

Figure 2. SUPERVISORY-LEVEL CONTROLLER THAT SETS A REFERENCE FOR LOW-LEVEL CONTROLLERS WITHOUT FEEDBACK OF THE SYSTEM

- Predictions of system behaviors over some future time interval (assuming some trajectory of control variables) are obtained using an explicit "internal model".
- Optimization of some aspects of system behavior over the future time interval is used to choose the control variable trajectory.
- Implementation of the optimized control trajectory includes only an initial segment of that trajectory. After that, the whole cycle of prediction and optimization is repeated over an interval of the same length. These prediction and optimization is performed online [19].

Standard online predictive control methods require full state feedback, which is not always feasible in practice (e.g. as is the case with a printer). Therefore, we introduce a method that makes use of simulation to *offline* pre-compute an optimal reference (whole optimized control trajectory) for the low-level controllers. Ideally, such a method would be used when it is not possible to use feedback of the system (e.g. when some parameters cannot be measured at runtime).

We assume that the pre-computation of the optimal input can be represented as a shortest path problem, which can efficiently be solved using graph search algorithms that are commonly used in routing: Dijkstra's algorithm [20] and the A* algorithm [21]. Dijkstra's algorithm and the A* algorithm have been chosen because they both find a global optimum (i.e. global optimal path).

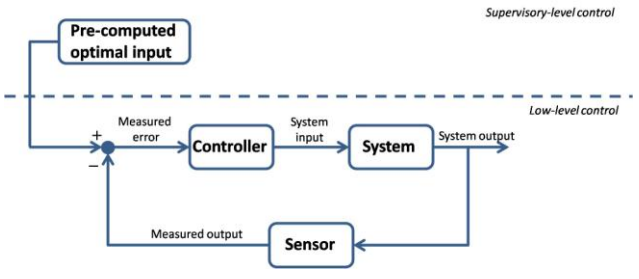

Figure 3. SUPERVISORY-LEVEL CONTROLLER WITH PRE-COMPUTED OPTIMAL INPUT FOR A LOW-LEVEL CONTROLLER

Copyright © 2010 by ASME

In this paper, the proposed method based on graph search algorithms is explained taking a printer as an example; the pre-computed system input for runtime state transition is derived using a Simulink model of a professional printer.

The paper is organized as follows. The next section, *Graph Theory and Shortest Path Problem*, gives an overview of Dijkstra's algorithm and the A* algorithm. In the section *State Transition*, we introduce a method that uses those two graph search algorithms to solve the state transition problem. The implementation of the method is presented in section *Case Study: Designing a Reconfigurable Printer*. Discussion and concluding remarks on the method and its validation are provided in the sections *Discussion* and *Conclusions*, respectively.

GRAPH THEORY AND SHORTEST PATH PROBLEM

Graph theory is the study of graphs – mathematical structures that are used to model relations between objects. Generally, graphs are used as an abstract representation of a set of objects and links that connect pairs of these objects (Fig. 4). A *graph* $G = (V, E)$ is composed of two sets $\{v_i\}$ and $\{e_{v_i v_j}\}$, where $\{v_i\}$ is a set of vertices (or nodes) formed by the objects and $\{e_{v_i v_j}\}$ is the set of edges (or arcs) between pairs of adjacent vertices. In this paper, we will consider weighted graphs, in which each edge e_{ij} has a transition cost c_{ij} (weight).

A *path* from v_S (source) to v_D (destination) is a sequence of vertices $(p_1, p_2, p_3, \dots, p_N)$ where $p_1 = v_S$ and $p_N = v_D$. Each path has a cost equal to the sum of the individual costs of the arcs $(c_{p_i p_{i+1}})$ in the path. An *optimal path* (or shortest path) from v_S to v_D is defined as a path that has the smallest cost over the set of all paths from v_i to v_j.

Finding shortest paths is generally done by searching a graph. There are various graph search algorithms available to find low-cost paths. In this work, we use Dijkstra's algorithm [19] and the A* algorithm [21].

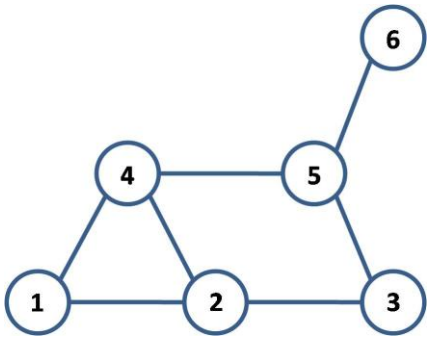

Figure 4. A GRAPH WITH 6 VERTICES AND 7 EDGES

Dijkstra's algorithm always finds the optimal path. Finding the shortest path, however, can take a long time especially if evaluating c_{ij} (by simulation) is computationally expensive. The A* algorithm is a method that tries to find an optimal path using fewer evaluations of c_{ij}. It will always find an optimal path if it uses a so-called admissible heuristic [21]. However, it also allows making a tradeoff between search time and solution quality [22]: using an inadmissible heuristic, the A* algorithm generally finds paths more quickly, but these paths may be suboptimal.

STATE TRANSITION

In theory there are an infinite number of possible paths between two states of the system. By only considering paths that go via a discrete set of substates (grid points), it becomes possible to use graph search algorithms to find the optimal path among all (discrete) possible ones.

The idea behind the proposed method is to optimize state transition (i.e. path that indicates the way adaptable variables of the system change) by minimizing the cost of that transition. Figure 5 presents an imaginary grid, vertex v_S that indicates the source state, vertex v_D that indicates the destination state, and two paths. The cost c_{SD1} of the purple path equals:

$$c_{SD1} = c_{V_{13}V_9} + c_{V_9V_5} + c_{V_5V_2} + c_{V_2V_3} + c_{V_3V_4} \quad (1)$$

The cost c_{SD2} of the blue path equals:

$$c_{SD2} = c_{V_{13}V_{10}} + c_{V_{10}V_{14}} + c_{V_{14}V_{11}} + c_{V_{11}V_8} + c_{V_8V_4} \quad (2)$$

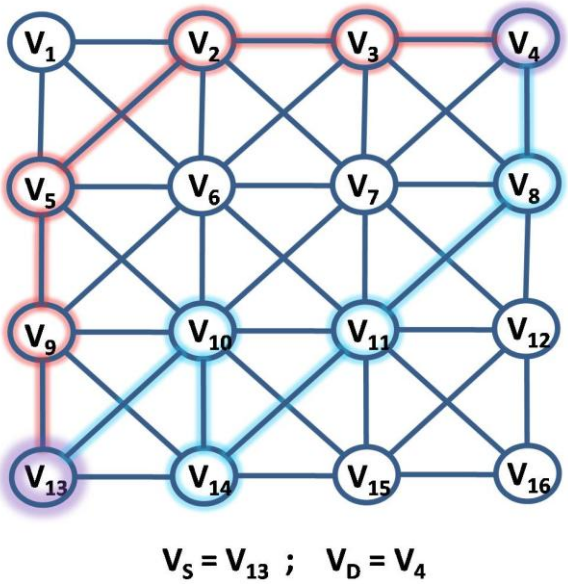

Figure 5. AN IMAGINARY GRID WITH TWO PATHS BETWEEN V_S AND V_D

Copyright © 2010 by ASME

The two paths presented in Fig. 5 are computed using two different cost functions. In case of a printer, one cost function may be related to the quality of the print while the second one may be related to the energy that is used to make the transition. It is also possible that the cost function considers both the quality of the print and the energy used to make the transition. In such a case, a tradeoff needs to be made.

Figure 2 shows supervisory-level controller with pre-computed optimal input for a low-level controller. To define the optimal input for the system, three steps need to be performed:

- Building a grid,
- Calculating costs between neighboring grid points,
- Applying an algorithm that finds the shortest path from source to destination.

The grid is built of a discrete set of substates and the transitions between them. Each vertex of the grid corresponds to a system state represented by a vector of adaptable input variables. The edges in the grid are associated to neighboring pairs of vertices. The neighboring vertices are the vertices between which transition is possible within one time unit. The source and the destination vertices are defined by the states between which the transition takes place (v_S and v_D).

Figure 6 shows an example of a steady-state approximation for computing transition costs from v_i to v_{i+1}. To compute the costs c_{ij}, we assume that the system has been in state v_i for an arbitrary long time (t_B). By doing so, we make an approximation of the state transition costs not taking into account the actual "history" of the system. After time t_B, a state transition takes place. That state transition ends at time t_E. After that time, the system operates in state v_{i+1}. As presented in the Fig. 6, the cost of the state transition is calculated only over the time interval $[t_B, t_E]$.

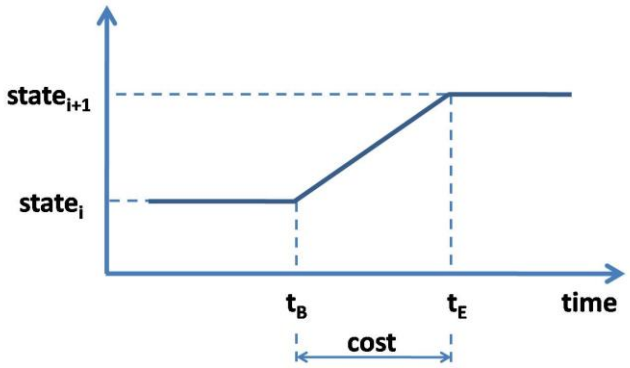

Figure 6. AN EXAMPLE OF STEADY-STATE APPROXIMATION FOR COMPUTING STATE TRANSITION COSTS

When all transition costs are computed, they are used as input for Dijkstra's algorithm or the A* algorithm and the shortest path is computed. After that, the obtained shortest path is used as the pre-computed optimal input for the system.

CASE STUDY: DESIGNING A RECONFIGURABLE PRINTER

To illustrate the implementation and applicability of the method, we have applied it to the design of a reconfigurable printer. Within Océ-Technologies B.V., a physical model of the printer system has been developed using Simulink [23]. That model defines four adaptable input variables: temperature T during printing, speed V of printing, latitude L that determines a "safety margin" for T and V to guarantee high-quality printing, and paper mass m_{pap}. A reconfigurable printer should be able to go from a state (L_S, T_S, V_S, m_{papS}) to a state (L_D, T_D, V_D, m_{papD}) at runtime. During such a transition, the output quality measure $Q(t)$ should be within a range $(-q, q)$. For confidentiality reasons, the actual values of $L_S, L_D, T_S, T_D, V_S, V_D$, and q cannot be given.

We have chosen to use a regular, uniform grid. The resolution of the grid was set to:

$$GR = L_{grid} \times T_{grid} \times V_{grid} \times m_{grid}. \quad (3)$$

Grid edges e_{ij} for which $sign(v_j - v_i)$ was opposite to $sign(v_D - v_S)$ for one or more dimensions were excluded. It was assumed that only two kinds of paper are used, 80 g/m^2 and 200 g/m^2. Since paper mass cannot be varied continuously, a step function was used instead of the ramp shown in Fig. 6.

The cost function was not a part of the system. Therefore, it had to be properly introduced in the simulation model. The cost function was defined as follows:

$$c_{estimated} = \int_{t=t_B}^{t_E} |Q(t)|^{\beta} \, dt \quad (4)$$

where $Q(t)$ is a measure of print quality, which depends on the relation between the speed of transferring the toner to the paper, the temperature of both the toner and the paper, and the pressure applied while transferring the toner to the paper. Smaller values indicate better quality. β is a user-defined parameter.

As shown in Fig. 6, the cost of the state transition was calculated only over the time $t_E - t_B$. In this study, t_B was set at 500 seconds. $t_E - t_B$ was set to $\alpha \cdot |v_j - v_i|_0$, where $|...|_0$ denotes the L_0-norm. In this way, each possible path from v_S to v_D takes the same time. α is a user-defined parameter. The minimum transition time was chosen such that there is no over- and/or undershooting.

To illustrate some characteristics of the method, various scenarios to compute optimal input for the system have been used. These scenarios are described in Tab. 1. When β is set to

Copyright © 2010 by ASME

0, the cost is simply the length of the time interval of the integration function. In each of these scenarios, Dijkstra's algorithm was used to find the shortest path. The values of α and β are varied.

RESULTS

Figure 7 shows typical results obtained when applying introduced method. The scenarios SC 9, SC 17, SC 12, and SC 20 (from top to bottom, respectively) have been selected to show the way the user-defined parameters α and β influence state transitions paths. Figure 7 presents the print quality (left) obtained after applying pre-computed optimal input (right) to the printer Simulink model. The range $(-q, q)$, which indicates when the output quality measure $Q(t)$ is optimal, is presented in the figure by two red dashed lines. The red stars in the "quality" figures indicate steps of state transitions.

Figure 7 shows that the number of transition steps was not the same for each scenario. For example, for SC 9, eight steps were needed to make the transition, while for SC 17, SC 12, and SC 20 only seven steps were needed.

Table 1. SCENARIOS TO COMPUTE IDEAL INPUT FOR THE SYSTEM

Scenario's name	α	β
SC 1	1	0
SC 2	2	0
SC 3	3	0
SC 4	4	0
SC 5	1	1
SC 6	2	1
SC 7	3	1
SC 8	4	1
SC 9	1	2
SC 10	2	2
SC 11	3	2
SC 12	4	2
SC 13	1	4
SC 14	2	4
SC 15	3	4
SC 16	4	4
SC 17	1	6
SC 18	2	6
SC 19	3	6
SC 20	4	6

The print quality curves are different for each scenario and its pre-computed optimal input. The effect of increasing β can clearly be observed. Higher values of β result in a stronger penalty for large values of Q. Increasing α results in more gradual changes of adaptable input variables (because $t_E - t_B$ increases). This results also in smaller absolute values of Q.

The results show that when α is small (e.g. α = 1), then β has a large impact on the cost and the optimal path (see Fig. 7, scenarios SC 9 and SC 17). When α is large (e.g. α = 4), however, it seems that β has lower impact on the cost and the optimal path (see Fig. 7, scenarios SC 12 and SC 20). Figure 7 shows an example when two different scenarios (i.e. SC 12 and SC 20) give different costs, but the optimal path for these two scenarios is exactly the same.

DISCUSSION

The goal of this paper was to communicate our experience in developing methods to manage system behavior using runtime state transitions. We have proposed an alternative approach to commonly used control theory. In our approach, the behavior of the system is handled using a pre-computed optimal input and not by a sensor that senses an output value and subtracts it from a desired value (like in control theory).

In the section *Case Study: Designing a Reconfigurable Printer*, we have shown an application of the method and some initial results. The proposed method, though simple, shows its potential and possible benefits that can be obtained when applying it to real systems. Although no history (dynamics) of the system was used to pre-compute optimal input, our method was able to handle dynamic system behavior. Nevertheless, further development of the method is required.

The method makes use of a steady-state approximation of a dynamic system. As a consequence, the computed costs are not necessarily correct. At the beginning, there is no (or small) effect of history on calculating the cost (e.g. from v_S (source) to v_{S+1}). Each further calculation of the cost is more and more effected by the history. In the future work, it will be investigated how the history of the system should be implemented in the introduced method. Moreover, it will be investigated whether taking history into account will have an effect on pre-computed optimal input.

For the clarity of the paper, the tests described here were designed such that i) changes made by adaptable variables (L, T, V, and m_{pap}) had to be performed as quickly as possible, and ii) time t_B was set at 500 seconds. It is, however, not known whether such approach is "optimal". It is, therefore, recommended to investigate the optimal size of the transition steps and the optimal time t_B.

The method was tested using a regular, uniform grid. Further work should include tests of the method when a grid with obstacles, representing forbidden system states, is used.

Copyright © 2010 by ASME

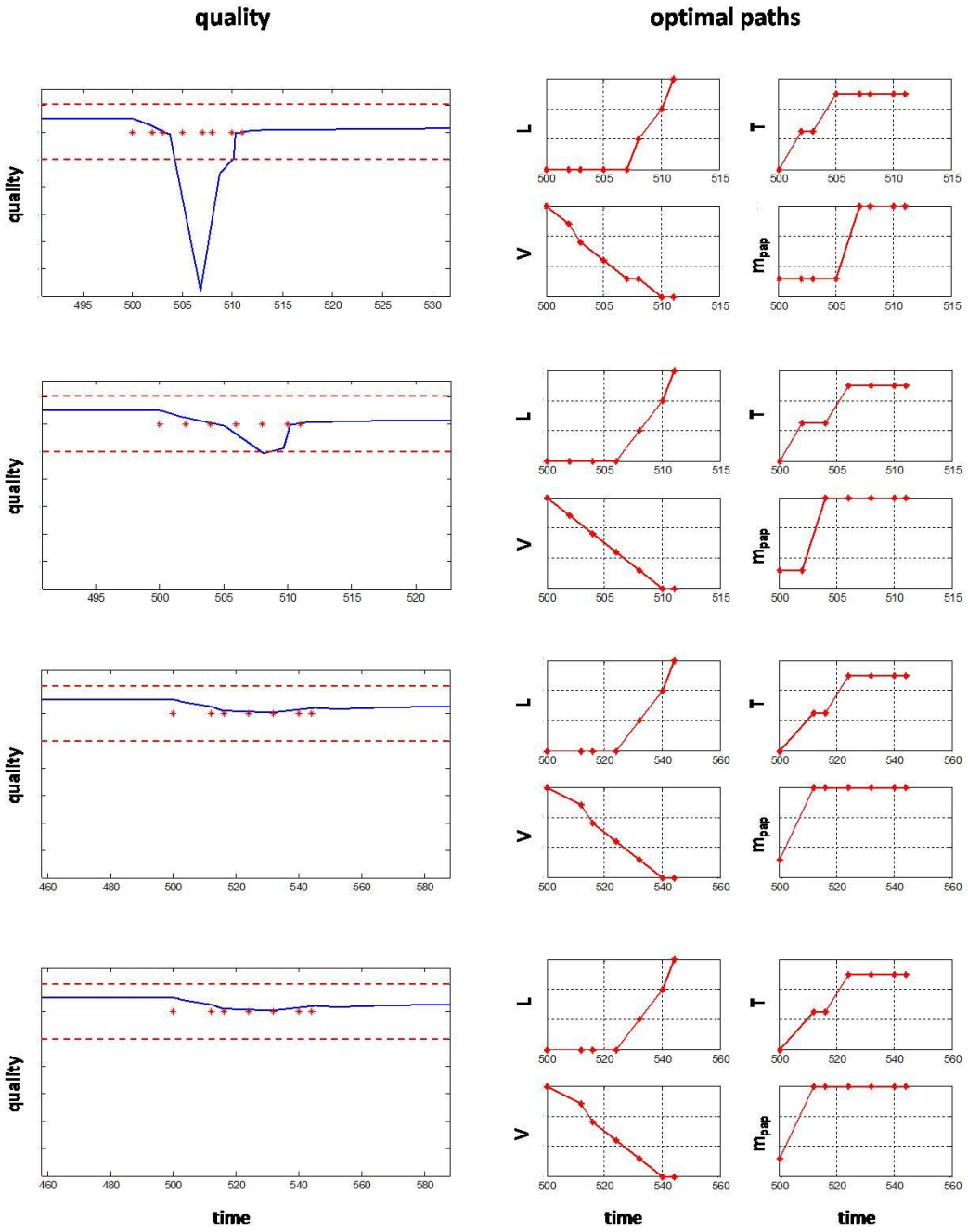

Figure 7. PRINT QUALITY (LEFT) AFTER APPLYING PRE-COMPUTED OPTIMAL INPUT FOR THE SYSTEM (RIGHT) (FROM TOP TO DOWN: SC 9 (α = 1, β = 2), SC 17 (α = 1, β = 6), SC 12 (α = 4, β = 2), AND SC 20 (α = 4, β = 6))

Copyright © 2010 by ASME

Another study should investigate whether using hierarchical grids [24],[25] is of benefit to achieve faster computation times when dealing with large grids. Also a method to define optimal resolution of the grid should be developed.

The pre-computed input was optimized for constant high-quality printing. The method, however, should be validated for different costs (e.g. energy used, a tradeoff between quality and energy used).

Pareto optimization is often used to deal with trade-offs in reconfigurable systems [26]-[29]. Theoretically, moving along the Pareto frontier results in optimal system performance. Ferguson and Lewis [30], however, reported that following the trajectory depicted by the Pareto frontier might be unsatisfactory when designing reconfigurable systems, since changes in configurations take time and development of a controller would be very complex. Our future research will investigate whether the method introduced in this paper can successfully be used to find optimal state transitions between two Pareto optimal points.

As mentioned in the *Introduction*, most of current research on reconfigurability focuses only on changes in the physical configuration of systems. The method presented in this paper was developed to handle dynamic system behavior. Further research is needed to investigate whether this method is able to handle both dynamic system behavior and physical reconfiguration of the system at (the same) runtime.

CONCLUSIONS

A new method to handle dynamic behavior of the system using a pre-computed optimal input has been developed. This method makes use of graph search algorithms that solve shortest path problems and that are commonly used in routing: Dijkstra's algorithm. The method was applied to the design a reconfigurable printer, which has to change its configuration to achieve maximum performance (i.e. high quality of print) when operating conditions are changing (e.g. speed of printing). A simulation study, of a new methodology has been conducted using a Simulink model. The method, though simple, shows its potential and possible benefits that can be obtained when applying it in real systems.

ACKNOWLEDGMENTS

We would like to thank Prof. Robert Babuška, of Delft University of Technology, and Carmen Cochior, of Eindhoven University of Technology, for useful comments during preparation of this paper.

This work has been carried out as part of the OCTOPUS project with Océ-Technologies B.V. under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Bsik program.

REFERENCES

[1] Weaver, J. M., Wood, K. L., and Jansen, D., 2008, "Transformation Facilitators: A Quantitative Analysis of Reconfigurable Products and Their Characteristics", *ASME International Design Engineering Technical Conference and Computers and Information in Engineering Conference*, New York, NY, DETC2008-49891.

[2] Olewnik, A., Brauen, T., Ferguson, S., and Lewis, K., 2004, "A Framework for Flexible Systems and Its Implementation in Multiattribute Decision Making". *ASME Journal of Mechanical Design*, 126(3): 412-419.

[3] Ferguson, S., Tilstra, A. H., Seepersad, C. C., and Wood, K. L., 2009, "Development of a Changeable Airfoil Optimization Model for Use in the Multidisciplinary Design of Unmanned Aerial Vehicles," *ASME International Design Engineering Technical Conference and Computers and Information in Engineering Conference*, San Diego, California, DETC2009-87482.

[4] Subramanian, N., and Chung, L., 2001. "Software architecture adaptability: an NFR approach", *Proceedings on the 4th International Workshop on Principles of Software Evolution*. Vienna, Austria, pp. 52-61.

[5] Gu, P., Hashemian, M., and Nee, A. Y. C., 2004, "Adaptable design". *CIRP annals, International Institution for Production Engineering*, 53: 539-557.

[6] Chung, L. and Subramanian, N., 2004, "Adaptable architecture generation for embedded systems", *Journal of Systems and Software*, 71: 271-295.

[7] Olewnik, A. and Lewis, K., 2006, "A Decision Support Framework for Flexible System Design", *Journal of Engineering Design*, 17: 75-97.

[8] Opperman, R., 1994. "Adaptively supported adaptability", *International Journal of Human-Computer Studies*. 40: 455-472.

[9] Ferguson, S., Lewis, K., de Weck, O., and Siddiqi, A., 2007, "Flexible and Reconfigurable Systems: Nomenclature and Review," *2007 ASME DETC and CIE Conferences*, Las Vegas, NV, DETC2007/DAC-35745.

[10] Siddiqi, A., de Weck, O. L., and Iagnemma, K., 2006, "Reconfigurability in Planetary Surface Vehicles: Modeling Approaches and Case Study," *Journal of the British Interplanetary Society*, 59.

[11] Khire, R., and Messac, A., 2008, "Selection-Integrated Optimization (SIO) Methodology for Optimal Design of Adaptive Systems," *Journal of Mechanical Design*, 130(10): 101401.

[12] Olewnik, A., and Lewis, K., 2006, "A Decision Support Framework for Flexible System Design," *Journal of Engineering Design*, 17(1): 75-97.

[13] Skiles, S. M., Singh, V., Krager, J., Seepersad, C. C., Wood, K. L., and Jensen, D., 2006, "Adapted Generation and Computational Techniques for the Application of a Transformer Deign Theory," *ASME International Design Engineering Technical Conference and Computers and*

 Copyright © 2010 by ASME

Information in Engineering Conference, Philadelphia, Pennsylvania, DETC2006-99584.

[14] Singh, V., Walther, B., Koraishy, B., Krager, J., Wood, K. L., Putnam, N., and Jensen, D., 2007, "Design for Transformation: Theory, Method and Application", *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Las Vegas, NV, DETC2007-34876.

[15] Chmarra, M. K., Arts, L., and Tomiyama, T., 2008, "Towards Adaptable Architecture," *ASME International Design Engineering Technical Conference and Computers and Information in Engineering Conference*, New York, NY, DETC2008-49971.

[16] Arts, L., Chmarra, M. K., and Tomiyama, T., 2008, "Modularization Method for Adaptable Products," *ASME International Design Engineering Technical Conference and Computers and Information in Engineering Conference*, New York, NY, DETC2008-49338.

[17] Patton, R.J., Frank, P.M., and Clark, R.N., 2000, Issues of Fault Diagnosis for Dynamic Systems, Springer-Verlag, London, April 2000.

[18] Franklin, G. F., Powell, J. D., and Emami-Naeini, A., 1993, Feedback Control of Dynamic Systems, Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA.

[19] Maciejowski, J.M., 2002, Predictive control with constraints, UK: Prentice Hall.

[20] Dijkstra, E.W., 1959, "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik*, 1: 269–271.

[21] Hart, P.E., Nilsson, N.J., and Raphael, B., 1968, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Transactions of Systems Science and Cybernetics*, 4(2): 100–107.

[22] Davis, H. W., Bramanti-Gregor, A., and Wang, J., 1989, "The advantages of using depth and breadth components in heuristic search", *Methodologies for Intelligent Systems*, 3: 19-28.

[23] MATLAB, "The MathWorks Inc." 2008, version R2008A.

[24] Car, A., and Frank, A. U., 1994, "Modelling a Hierarchy of Space Applied to Large Road Networks", *Lecture Notes in Computer Science*, 884: 15–24.

[25] Zhao, Y., 1997, Vehicle Location and Navigation Systems. Artech House Inc., Boston, Massachusetts, United States.

[26] Pareto, V., 1906, "Manuale di economica polittica", Societa editrice libraia, Milan, Italy, 1906, translated by Schwier AS (1971) Manual of political economy. Macmillan, New York, pp 259–262.

[27] Guesnerie, R., 1975, "Pareto optimality in non-convex economies", *Econometrica*. 43 1-29.

[28] Kacem, I., Hammadi, S., and Borne, P., 2002, "Pareto optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic", *Mathematics and Computers in Simulation*. 60: 245-276.

[29] Baumgartner, U., Magele, Ch., and Renhart, W., 2004, "Pareto optimality and particle swarm optimization", *IEEE Transactions on Magnetics*. 40: 1172-1175.

[30] Ferguson, S., and Lewis, K., 2006, "Effective development of reconfigurable systems using linear state-feedback control", *AIAA Journal*. 44: 868-878.

Copyright © 2010 by ASME