OPTIMAL PLANNING IN THE REAL WORLD

Wayne J. Davis
Professor Emeritus
Industrial and Enterprise Systems Engineering
University of Illinois at Urbana-Champaign

Abstract

This paper questions what utility, if any, optimal planning might provide in managing real-world systems. Using a simple production planning problem, the paper first explores the relationship between traditional planning over a temporal horizon and the classic two-point, boundary-value optimal control. It then extends these relationships throughout a multi-level manufacturing system. Building upon these explorations, the paper introduces a new paradigm for systems engineering. This introduction previews how the three principal system technologies – planning, control, and mechanics – might be unified under the proposed paradigm. The latter discussion also introduces a second temporal upon which these three unified technologies might be addressed concurrently in real time.

INTRODUCTION

The conventional planning problem consists of two basic components -- a set of feasible alternatives and a performance measure for evaluating the utility of each alternative. The conventional planning process consists of two steps -- the formulation of the planning problem and its subsequent solution. From such a perspective, planning could be characterized as a task, particularly since implementing the solution is seldom addressed. The need to monitor the plan’s continued viability and the associated need for periodic revisiting the planning task has been discussed for certain applications. Even in such scenarios, the relationship between the plan’s implementation and assessment is neglected. Consequently, planning becomes reactive rather than proactive. That is, planning is trying to keep pace with the real world rather than managing what occurs in the real world. One must ask the question, “What, if any, benefits might this traditional planning approach provide towards managing outcomes in the real world?”

This paper explores this question by introducing a trivial production planning problem, and then reformulating the same problem as an optimal control problem. By providing both perspectives, it is shown that the production planning problem, like most temporal planning problems, represents an instance of the classic two-point, boundary value (TPBV) problem. Specifying the TPBV problem requires one first to define the state transition mechanism that governs the considered system’s dynamics, the particular state from which the defined state trajectory initiates, and the desired final state to be approached.

Upon further consideration of the TPBV problem, one concludes that it is impossible to plan from the present. That is, the TPBV planning must consider a system’s dynamics between two future states. The inherent difficulties and uncertainties associated with planning between two future states for the planning task is addressed. In
particular, it is demonstrated that the planner never has full control over which states might or should occur. Rather, the specification of these future states must account for the interactions of the given system with other systems in the real world.

A multi-level manufacturing system is then introduced and the functional relationships among its component subsystems are outlined. The variety of potential behaviors for characterizing the dynamics of these component systems is also discussed. Particular attention is given to the discrete-event dynamics associated with execution of tasks.

The conclusion poses several questions. Should one seek an optimal plan in the real world? Can plans be optimally implemented in the real world? Can or should real-world planning be addressed as an individual activity? If not, how might one characterize and approach collaborative planning among interacting subsystems?

**A PRODUCTION PLANNING AND CONTROL PROBLEM**

Consider a simple production planning for the discrete times \( t_k \) between the initial time \( t_{initial} \) and the final time \( t_{final} \). If these discrete times partition the interval \([t_{initial}, t_{final}]\) into \( K \) uniform subintervals, then

\[
\begin{align*}
  t_k &= t_{initial} + k\Delta \\
  \Delta &= \frac{t_{final} - t_{initial}}{K}
\end{align*}
\]

where

\[
\begin{align*}
  \Delta &= \frac{t_{final} - t_{initial}}{K} \\
  t_0 &= t_{initial} \\
  t_K &= t_{final}
\end{align*}
\]

Assume that the production during any period \( p(t_k) \) is capped

\[
p(t_k) \leq p_{max} \quad (k = 1, \ldots, K)
\]

Let \( d(t_k) \) denote the forecasted demand for the considered product during the interval \( (t_{k-1}, t_k] \). The producer seeks to satisfy each period’s demand with sales \( s(t_k) \). However, a given period’s sales cannot exceed its demand, or

\[
s(t_k) \leq d(t_k) \quad (k = 1, \ldots, K)
\]

Because the producer operates in a competitive market, any unfulfilled demand in any period is forfeited.

The sales in each period is also limited by the sum of each period’s production \( p(t_k) \) and the remaining inventory at the end of the last period \( i(t_{k-1}) \), or

\[
s(t_k) \leq i(t_{k-1}) + p(t_k) \quad (k = 1, \ldots, K)
\]

*Symposium on Complex Systems Engineering: Putting Complex Systems to Work*
Any unsold product can be carried over to the next period, giving

\[ i(t_k) = i(t_{k-1}) + p(t_k) - s(t_k) \quad (k = 1, \ldots, K) \]  

(7)

Space limitations impose an upper limit on the remaining inventory at the end of each period, or

\[ i(t_k) \leq i_{\text{max}} \quad (k = 1, \ldots, K) \]  

(8)

If a linear objective function is specified for these constraints, the resulting problem can be solved as a linear program. Before formulating that objective function, however, let us reconsider the production planning problem from a system perspective.

Traditionally, system models describe the temporal evolution of the state variables, whose definition is dependent upon the boundary that the modeler arbitrarily employs to distinguish the system from its environment. Physically, however, the system is seldom isolated from its environment; the system continues to interact with its environment.

Typically, one assumes that the system’s state \( x(t) \) evolves in a continuous manner such that its state transition function can be expressed via the differential equation

\[ \frac{\partial x(t)}{\partial t} = f(x(t), u(t)) \]  

(9)

where \( u(t) \) represents the vectors of the interactions between the system and its environment at time \( t \). Because only the simplest differential equations can be solved analytically, the state transition function is often discretized as

\[ x(t_k) = f^A(x(t_{k-1}), u(t_{k-1})) \]  

(10)

and numerically integrated upon digital computers. If the considered system is linear, its discrete state transition function can be simply expressed as

\[ x(t_k) = A^\Delta x(t_{k-1}) + B^\Delta u(t_{k-1}) \]  

(11)

Assume that the interaction vector \( u(t_k) \) can be decomposed into two component vectors – the control inputs \( c(t_k) \) and the external disturbances \( d(t_k) \). The state transition function can then be rewritten as

\[ x(t_k) = A^\Delta x(t_{k-1}) + B^c c(t_{k-1}) + B^d d(t_{k-1}) \]  

(12)

Returning to the derived equality for our production system

\[ i(t_k) = i(t_{k-1}) + p(t_k) - s(t_k) \]

one can define the state vector to be the inventory
\[ x(t_k) = \begin{bmatrix} i(t_k) \end{bmatrix} \]  

and the control vector to be the production and sales,

\[ c(t_k) = \begin{bmatrix} p(t_k) \\ s(t_k) \end{bmatrix} \]  

The inventory equality can then be expressed as the state transition function where

\[ x(t_k) = [1] x(t_{k-1}) + [1 \quad -1] c(t_k) \]  

with

\[ A^\Delta = [1] \text{ and } B^\Delta = [1 \quad -1] \]  

Observe that the demand does not influence the state transition mechanism per se; rather, the demand constrains the sales component in the control vector (see Eq. 5). Either an open-loop or closed-loop strategy might be adopted in the computation and subsequent implementation of the control vector \( c(t_k) \). This paper will discuss the open-loop strategy where the control vector \( c(t_k) \) for the discrete times \( t_k \quad (k = 1, \ldots, K) \) is specified on an a priori basis by solving an optimal control or planning problem.

The considered optimal control problem might take several forms. Perhaps the most basic problem is the two-point, boundary-value (TPBV) problem:

Given the initial state \( x(t_0) \) and the forecasted disturbances \( d(t_k) \quad (k = 1, \ldots, K) \), compute the optimal control sequence \( c^*(t_k) \quad (k = 1, \ldots, K) \) which solves

\[ \min \delta(x(t_K) - x^d(t_K)) \]  

s.t. \( x(t_k) = f^\Delta(x(t_{k-1}), c(t_k), d(t_k)) \quad (k = 1, \ldots, K) \)  

The objective function in problem P1 penalizes any deviation between the optimal final state \( x^*(t_K) \) and the desired final state \( x^d(t_K) \). If the desired final state \( x^d(t_K) \) can be reached from the specified initial state \( x(t_0) \) in \( K \) steps, the optimal value of the penalty function is zero. However, if \( x^d(t_k) \) cannot be reached, the optimal value of the penalty function increases monotonically as distance of the optimal final state \( x^*(t_K) \) from the desired final state \( x^d(t_K) \) increases.

The ability to reach the desired future state from a prior state constitutes the classical controllability problem. If a linear system with \( n \) state variables is controllable, then one theoretically should be able to transfer between any two states in \( n \) steps. However, this assertion assumes there no constraints are imposed upon the control inputs, the states that can be traversed in reaching the desired final state, or upon the duration
required to reach the desired final state. Physically, such constraints exist and must be considered.

It is impossible to make similar generalizations regarding controllability for nonlinear or constrained systems. Perhaps a more useful concept is that of reachability, which defines a set of states $X(x(t_0), t)$ that can be reached from the initial state $x(t_0)$ by time $t (> t_0)$. Returning to the TPBV problem, the optimal final state $x^*(t_K)$ can equal the desired final state $x^d(t_K)$ only if the desired final state is reachable, or

$$
x^d(t_K) \in X(x(t_0), t_K)
$$

Otherwise, the optimal final state will approach the desired final state to the extent that it is possible.

The objective function in P1 does not consider any costs associated with implementing the optimal control sequence $c^*(t_k) \ (k = 0, \ldots, K - 1)$. Most real-world problems require a compromise between the cost of implementing the selected control sequence and the cost of deviating from the desired final state. In that regard, the basic TPBV problem might be restated as

$$
\text{min } \sum_{k=0}^{K-1} \text{cost}(c(t_k)) + \delta(x(t_K) - x^d(t_K))

\text{ s.t. } \ x(t_k) = f^*_k(x(t_{k-1}), c(t_k), d(t_k)) \ (k = 1, \ldots, K)
$$

P2

Whenever the cost of implementing the control input is considered, the only way one can guarantee that the final state will equal the desired final state is to impose that requirement as an explicit constraint. However, imposing such a constraint might render problem P2 infeasible should $x^d(t_K) \notin X(x(t_0), t_K)$. Later we will observe that the two terms in the considered objective function correspond to two distinct roles for the planner.

For our production planning example, the problem P2 can be stated as

$$
\text{min } p \sum_{k=1}^{K} \ r_k(t_k) - s \sum_{k=1}^{K} \ s_k(t_k) + \delta(I(t_k) - I^d(t_K))

\text{ s.t. } \ I(t_k) = I(t_{k-1}) + P(t_k) - S(t_k) \ (k = 1, \ldots, K)

\text{ P3}

\text{ I(t_0) given}

One immediately observes that if the selling price $s$ is greater than the production cost $p$, the problem becomes unbounded (has no finite minimum) because no constraints have been imposed upon the control inputs. Including the previously derived constraints upon the states and control inputs to our production planning problem, result in the following TPBV problem

---

*Symposium on Complex Systems Engineering: Putting Complex Systems to Work*
Traditionally, the planner is not responsible for implementing the planned control sequence. In fact, the planning literature generally ignores implementation issues. There is one important consideration that cannot be ignored when any planner interacts with the real world; the planning horizon cannot include the present. If a planner must ponder an immediate response, the planner forfeits any opportunity to influence the present. Therefore, realistic planning horizons must span two future times. Consequently, the planned response must initiate from what might be the system’s state at the initial time and approach a moving target that resides even further within the system’s uncertain future.

Planning problems can seldom be precisely specified or solved given these inherent uncertainties. Realistically, any derived solution necessarily addresses a scenario that will never occur. One might question the inherent futility of specifying the solution to a scenario that will never occur. Some might contend that the planner has no other alternative. Solving for something is obviously better than solving for nothing. Or is it?

Let us consider scheduling, which has seldom been questioned in the literature. The typical scheduling problem addresses the future response of an artificially specified subsystem within a larger system. In order to debunk the intrinsic benefits of scheduling, one need only provide a single situation where implementing the proposed optimal schedule will be deleterious either to the performance of the considered subsystem or the larger system within which it is imbedded. Unfortunately, such examples are typical.

A new conceptual approach to planning is needed. Traditionally, planning has been addressed as a task, where one first formulates the planning problem and then solves it. Each planning task terminates whenever an optimal solution is obtained. Since implementing the solution is seldom considered, the planner usually waits until the next planning task (i.e. replanning) is initiated. Suppose planning could be addressed as a process where future plans were continuously updated even as the elements of prior plans were being implemented. The considered problem would be constantly refined implying the process of planning would be ongoing. Maintaining an optimal solution to a dynamic problem would probably be futile, if not impossible. A new planning philosophy must be formulated.

**TOWARD DISTRIBUTED, ON-LINE PLANNING**

This section explores planning as a process via the production planning hierarchy pictured in Figure 1. At the middle level of the hierarchy, the planned state or inventory
trajectories for the next eight weeks are depicted for three products – Red, Green and Blue. These planned responses extend beyond the prior observed responses to the right of the instantaneous origin or the present. As discussed above, however, the inventory manager’s planning horizon must not include the present. Rather, it must span two future times. Let’s assume that the inventory manager’s planning horizon spans from the end of week one to the end of week eight. This assumption implies that the production plan for the current week must have been determined and is currently being implemented by another agent.

In Figure 1, the implementing agent, or the cell manager, plans a more detailed response for manufacturing the Red, Green and Blue products during week one. The manufacturing of three units of each product is projected along a distinct axis immediately below the inventory manager’s trajectory. As shown, the manufacturing of the first unit of each product was initiated approximately half a week ago. The production of the next two units of each product should occur during the next production week.

![Figure 1- State Trajectories for a Multilevel Production Planning System](image-url)

---

*Symposium on Complex Systems Engineering: Putting Complex Systems to Work*
The prior manufacturing trajectory of the first item of each product is represented by a rectangle of unit height that spans the interval between its initial and final manufacturing times. This spanning rectangle is then further subdivided into smaller rectangles. The solid rectangles depict time intervals where the control of manufacturing the given item has been delegated to one of three machining centers. The open rectangles, on the other hand, indicate subintervals when the cell manager either queues or orchestrates the transport of the given item among workstations.

The planned manufacturing trajectories to the right of the present origin represent the probability that the associated item is being manufactured at a given future moment. At its left, each unit’s trajectory monotonically increases toward one, depicting the probability that the associated unit’s production will have been initiated. At its right, the trajectory monotonically decreases to zero as the probability of the unit’s production declines.

Below the cell manager’s trajectories, the state trajectory for each of three machining centers is given. The solid connecting arcs of a given color between the various state trajectories depict when the associated unit is physically being transported between two manufacturing centers. A dashed line, on the other hand, depicts an empty transporter moving toward a given station in order to pick up an item of a that color.

The controller at a given machining center manages each unit while it resides at the center. The solid rectangle represents when the unit is actually being processed, and the open rectangle depicts when the product resides in the machining center’s queue.

The machining managers routinely project the planned manufacturing trajectories for the units that currently reside within their control domain. The cell controller fuses the current projection from the machining centers to specify the probabilistic future state from which it will initiate its planning for the next two items of each product.

In order to appreciate the dynamic behavior of the proposed interaction better, consider the following example. Suppose that a controller manages a multi-threaded computer upon which several tasks have been initiated. The first task might be downloading a large file from the web. The second might be printing a large file while the last task might be executing a large program. With respect to the first task, a status window might depict the size of the file being downloaded, the remaining file sectors to be downloaded, the average rate at which data is being transferred, and the remaining time to complete the download task. The printer status window might depict status of the printer, the original number of pages to be printed, and the remaining number of pages to be printed.

With respect to the third task, the executing code would likely provide a status window for depicting its progress. The computer manager cannot control the time that each task requires. It can, however, prioritize the resource allocation for each task. More importantly, it can monitor the progress toward executing the current tasks as it schedules subsequent tasks. The existence of subsequent tasks further implies that the computer manager addresses other more comprehensive tasks or goals.

Perhaps we should also spend a moment discussing the discrete natures of the individual tasks. Traditionally, systems analyses address continuous systems whose dynamics are governed by differential equations. As mentioned earlier, most differential equations are now routinely discretized for digital computation.
The downloading of a file approximates a continuous phenomenon even though the data arrives frequently in small discrete packets. When the requested file has been downloaded, however, the task terminates. Like most managed tasks, the downloading task is punctuated with a beginning and an end, and the continuous behavior occurs only between these events. Moreover, that continuous behavior need not be similar for the repeated execution of a particular task. For example, the data might be delivered at a different rate when the next file is downloaded. Additional discrete events might also occur, which temporarily interrupt the data transmission.

The printing task includes other discrete phenomena, including several discrete subtasks. The file must first be downloaded to the printer. If the printer is in standby mode, it must be readied for printing. Each page of the document is typically printed as an individual entity, and the printing of an individual page can further be decomposed into other tasks.

Finally, the execution of a computer program provides yet another discrete behavior. The compiled program represents sequences of machine instructions that are invoked whenever a particular situation occurs. The execution of these instructions is synchronized with the processor’s clock. If the program executes offline, one might be able to track its progress toward its completion. Such characterizations are more difficult for the online execution of a program. For example, as this paper was keyed, each keystroke triggers the execution of a set of instructions. Some instructions buffer the character into a file, others display it upon the screen. Still others record the line and column where the character will appear on the finished page. If the character is the spacebar, another word is added to the word count while the spelling of the entered word is being checked.

The dynamics of the above manufacturing example are clearly unique. Each product requires the sequential execution tasks at the various machining centers. Before any task can be executed, however, the product must be configured into a proper state. This might include the completion of prior tasks, require delivering the product to the appropriate machining center, require additional components to be delivered to the same machining center to support the execution of the task, or it might require the product to wait until the machine completes its current task.

In the past, discrete-event simulation packages have often been employed to model the discrete dynamics of such systems. However, the lack of a formal description of such systems created a situation where one simulation package might predict different dynamics than another. Davis [1,2] drafted a formal definition for the state transitions of such systems, but certainly more development is needed. The discrete characteristics of most systems’ dynamics must be addressed, particularly when sequences actions are involved.

Let us review the nature of planning associated with the various systems by returning to Figure 1. There are minimal planning opportunities at the machine center level. Typically, the tasks will be assigned to and executed by the machining center in serial order. As the machining center manages the execution of its assigned tasks, it constantly projects when each task will be completed. Each machining manager then forwards its projections to the cell manager where they are fused into the initial future state from which the cell manager initiates its planning. In a recursive fashion, the cell
manager executes its assigned tasks. Typically, these tasks are composed of several component tasks, which will be reassigned to its managed resources for execution.

The cell manager’s planning involves defining which resource will execute each component task and in what order. One might formulate the cell manager’s planning problem as an integer programming problem. Unfortunately, the complexity of such formulations often precludes the consideration of the phenomena that support the task execution process. One can only imagine the constraints required to permit the cell manager to sequence the material transfer in Figure 1. Employing integer programs also ignores the stochastic nature of the task execution process.

Davis [3] advocates a real-time planning approach that relies heavily upon on-line simulation technologies. Unfortunately, on-line simulation technologies have received minimal attention. Development of on-line planning algorithms have received even less attention.

The cell manager receives its assigned tasks for the current week from the inventory manager. Earlier we designated each week’s production as a control variable. The inventory manager initializes its planned response from the cell manager’s projected outcome for the first week. The cell manager’s planning establishes the production and sales during the next seven weeks. In solving its TPBV problem, the inventory manager maximizes its profits throughout the next seven weeks. It then forwards these projected profits to a financial manager that establishes profit goals for the next 6 months. The estimated profits from the inventory manager are employed to initialize the first two months in the financial manager’s planning. Since the profits for the second month reside in the future, the financial manager’s plan can consider months two through six.

The actual interactions among the managers can be more complex than discussed here. As the machining controllers update their status in real time, the cell manager and machining managers mutually reassess the viability of the mutual production goals. The cell manager might also query a given machine manager regarding its ability to execute a particular task before assigning the task. In a recursive fashion, the cell manager continuously interacts with the inventory manager to reassess the viability of their mutual goals. Similarly, the inventory manager continuously interacts with the financial manager in order to establish meaningful goals for future profits.

In short, each manager constantly reviews its future plans in real time. Certainly, formulating a control strategy for taking the managed elements from a projected initial future state to a desired later final state is a critical element of the manager’s planning. However, the collaborative specification of the initial planning state and governing goals are equally critical. The manager cannot impose an infeasible goal upon any managed agent without creating an uncontrollable situation. Therefore, the manager must rely upon its managed agents to validate any goal assignment and to project its expected outcome in executing its assignment. Each manager is also managed. In this role, the managed manager seeks realistic assignments to govern its future response while providing continuous projections of its future response to its managing managers.

So what contribution does optimality provide in this situation? Unfortunately it contributes very little; and it may be futile if not detrimental. Traditional planning with its quest for optimality isolates the planner from the systems with which it interacts. It assumes deterministic values for future interactions and further assumes that its planned control trajectory can be translated into executable goals for its managed agents.
Moreover, traditional planning assumes that an accurate model can be specified for the interactive response of its managed agents. None of these assumptions is realistic; and unfortunately, the validation of optimal planning is critically dependent upon the validity of these assumptions.

LOOKING AHEAD

Assessing optimization’s role in real-world planning is only one of many issues that I encountered while seeking a new system’s paradigm by conceptually unifying the three core system technologies – mechanics, controls and planning. Critical assumptions were questioned in each system technology. Eventually, the inherent limitations toward observing behaviors in the real world were scrutinized. In fact, the new paradigm assumes that present in neither observable or controllable. It employs a second temporal axis, the virtual time axis, whose origin centers about the present moment in real-time. The present is then characterized upon the virtual time axis as the interval between two open sets – the past (to the left of the virtual origin) and the future (to the right of the virtual origin). Given the scope of this effort, it is virtually impossible to document this new paradigm within a single paper.

The New Paradigm in a Nutshell

The remaining paper briefly introduces the new paradigm using the above production planning example. Figure 2 presents modified state trajectories of the multilevel production planning system. Earlier, we observed that temporal planning must span two future times; one cannot plan for the present. In Figure 2, only the included machines are allowed to touch real time. A dedicated manager controls the imminent behavior of each machine just before it is physically instantiated into the immediate past. The machine’s manager never touches the present, however. It can manage the future and observe the present. As it addresses the future, each manager routinely projects the machine’s future trajectory while executing its current tasks.

In Figure 2, the cell manager is further displaced from the present. It derives no utility from feedback information about the machine’s current state. Rather the cell manager relies upon the projected future state of each machine upon completing its current tasks. The cell manager employs the projected completed state from each machine manager to initialize the planning for executing its assigned tasks. In a recursive fashion, the cell manager’s projection response provides the feedforward information, which the inventory manager then employs to initialize its planning. Although Figure 2 terminates at the financial manager level, this recursive behavior allows planning to be extended as far into the future as one desires.

The modified state trajectories in Figure 2 highlight a fundamental misconception regarding the existence of and the relationship among the component systems within a complex system. Looking at either Figure 1 or 2, one might conclude that the production planning system is hierarchical, implying some of the subsystems are subordinate to the others. Unfortunately, characterizing some of the systems as being controllers or managers furthers this impression. The new paradigm, however, no longer characterizes...
the systems via subordinate relationships. Conventional descriptors such as hierarchical or autonomous are irrelevant.

The new paradigm employs the notion of containment to characterize the relationships among the included systems. In Figure 2, a dedicated machine manager cannot exist independently of the machine it manages. Moreover, its projected response extends beyond the best estimate of the machines present state. Similarly, the response of the cell manager initializes from the completion of each machine manager.

The new paradigm also rejects the simplistic characterization of large systems as systems-of-systems. In the above system, only the machines exist as distinct components. The other managers cannot exist apart from the subsystems, which they contain. Perhaps, characterizing these large systems as systems-within-systems might be more appropriate.

The new paradigm employs two types of objects. The first type of object, a process, represents a physical entity whose state evolves in the real world. The second type of the object is a coordinator that either manages the behavior of process or other controllers. In Figure 3(a), the thick vertical line represents the current time. The past and future resides to the left and right of the vertical time line, respectively. The process is represented by the blue dot that resides upon the current time (vertical) line, designating that its physical state evolves within the real world and in real time. More
precisely, the process’s physical state evolves by instantiating a potential imminent state into the most immediate past state. This instantiation proceeds in real-time so long as the process physically exists. The blue horizontal line to the left of blue dot represents the machine’s prior trajectory.

Figure 3(b) depicts the projected trajectory of the dedicated machine manager. At the left end, a blue arc partially encircles the current state of its dedicated machine. This arc couples the initial state of the projected trajectory of the dedicated controller to the current state of its dedicated machine. Observe that the arc bridges, but does not touch, real time. The process manager observes the process’s immediate past as it seeks to
influence the machine’s imminent future while the process presently instantiates a potential imminent state into its immediate past. A solid blue line to the right of this arc projects the process’s subsequent state trajectory beyond the machine’s current state. The projected trajectory terminates with a dot at the right end, representing the trajectory’s final. The projected final state resides upon thinner vertical line at the future time where it occurs.

Any intervening point along the blue state trajectory depicts the projected state at its associated future time. Because the projected state trajectory describes the state of a physical object, the object’s energy can be determined for every state along this projected trajectory. This observation simply acknowledges that the energy-based approaches such as Lagrangian and Hamiltonian provide the foundation to classical mechanics. Newton’s First Law states that an inertial response exists such that state trajectory progresses with constant energy. Newton’s Second Law describes how external forces acting upon the physical object can accelerate the object from its inertial response. The accelerating forces acting along these perturbations from the inertial response perform work upon the object and thereby increasing its energy. For a conservative system, the amount of work performed in transferring from the object’s initial state (the first arc) to its final state (the second arc) can be determined as the difference between the final and initial state’s energy.

Figure 3(c) projects the future trajectories of several multiple processes as they concurrently approach to their respective future states at a specified future time. The total work required in accomplishing these state transitions can be computed as the difference between the total final and initial energies. In Figure 3(d), the projected final states for the ensemble of physical objects are aggregated into a coupled response, which is then projected to the later final state at its termination. Observe that the representation of the aggregated process is identical to that of a controller connected to the current state of a process. The principal difference between the two projected trajectories is that the aggregated response cannot exist independent of the process controllers’ trajectories that it extends. That is, the projected aggregate response intrinsically contains the projected trajectories of the contained process and their dedicated controllers. In that regard, it is also important that the energy associated with the right-most aggregated final state, does not represent the energy of the aggregated final state per se. Rather, it represents the projected energy of the collective set of processes when their state trajectories reach the right-most future time.

Figure 3(e) expands 3(d) by including more processes whose aggregated trajectories are projected even further into the future. In order to emphasize the claim that the proposed structure is not hierarchical, one of the process’s trajectories has been assimilated into two distinct aggregated trajectories.

In Figure 3, the indicated future times are relative to the current time. The current time, on the other hand, continues to advance in real time. This implies that planning horizon for each must also advance with real time in order to insure the planning horizon always resides in the future. This, in turn, implies the specified initial and desired final states for planning each controller’s future trajectory must continuously evolve in real time. Consequently, each controller must address a dynamic TPBV planning. To address such dynamic problems, new on-line searches are being developed that mimic equilibration process underlying classical. The proposed equilibration-based searches
converge to a traditional optimization in the particular case where the initial and final states are fixed.

While formulating the new paradigm, the need for an entirely new type of planning arose. Because the initial state for the aggregated response is dependent upon the final states of its contained trajectories, the interacting controllers cannot change their contributing state independently. Rather on-line collaborative planning algorithms must be developed to negotiate their coupled states in real time. While exploring potential collaborative search algorithms, a novel form of super symmetry has been discovered. It has long been recognized that primal and dual/adjoint formulations exist for the traditional or primal state transition mechanism. However, we never anticipated that each system also has a dual state transition mechanism that governs its dynamic interface with its contained systems for which primal and dual formulations also exist. Thus, each system has both a primal and dual configuration with each configuration having both primal and dual formulations.

A Concluding Remark

In his recent book, Simolin [4] devotes considerable effort to discussing the impact that prior unifications have made upon the scientific advancement. Although my recent discoveries toward unifying the basic system technologies are obviously less significant than the examples he cites, a plethora of potential research questions has emerged. The consequences toward my academic career were unexpected. Because the perceived unification negated several fundamental precepts across several technologies, I no longer felt comfortable teaching traditional courses. For example, should one teach scheduling when one believes that traditional scheduling is likely detrimental to larger system’s mission. Fortunately, I was able to retire and ponder my situation. After eighteen months of retirement, it is improbable that I will ever return to academia. Still the need to document my discoveries has not been forgotten. In that regard, I have prepared a set of narrated slides to provide an expanded introduction to the new paradigm. Perhaps, attending this conference might also provide insights regarding how I should proceed.

REFERENCES

4 Smolin, L. The Trouble with Physics (Houghton Mifflin, 2006).