An Agent-Based Framework for Fuel Cycle Simulation with Recycling

Matthew J. Gidden, Paul P.H. Wilson, Kathryn D. Huff, Robert W. Carlsen
Department of Nuclear Engineering & Engineering Physics, University of Wisconsin - Madison, Madison, WI, 53703
gidden@wisc.edu

Simulation of the nuclear fuel cycle is an established field with multiple players. Prior development work has utilized techniques such as system dynamics to provide a solution structure for the matching of supply and demand in these simulations. In general, however, simulation infrastructure development has occured in relatively closed circles, each effort having unique considerations as to the cases which are desired to be modeled. Accordingly, individual simulators tend to have their design decisions driven by specific use cases. Presented in this work is a proposed supply and demand matching algorithm that leverages the techniques of the well-studied field of mathematical programming. A generic approach is achieved by treating facilities as individual entities and actors in the supply-demand market which denote preferences amongst commodities. Using such a framework allows for varying levels of interaction fidelity, ranging from low-fidelity, quick solutions to high-fidelity solutions that model individual transactions (e.g. at the fuel-assembly level). The power of the technique is that it allows such flexibility while still treating the problem in a generic manner, encapsulating simulation engine design decisions in such a way that future simulation requirements can be relatively easily added when needed.

INTRODUCTION

Nuclear fuel cycle simulation is a practiced by many different nation states, academic institutions, and some private companies. The actors in this arena, however, often are interested in different categories of questions which tend to drive their simulation design decisions. The first simulators implemented over a decade ago were designed to answer broad reaching questions regarding the basic fuel stockpile makeups as a function of time given a variety of top-level fuel cycle decisions. For instance, one may investigate the relative repository requirements for different fuel cycles as a function of the aggregate elemental composition of their mass flows to the repository. Fairly quickly, however, one realizes that the granularity of the class of questions that can be investigate is rather coarse. Furthermore, for the majority of simulators, expert input is required to investigate even slightly different cases (e.g., investigating the effect of higher burnup fuels). Finally, because each simulator is championed by a different entity and developed in relative isolation, the actual simulation mechanics differ across the spectrum of simulators, causing benchmarking efforts to be cumbersome and making the simulators difficult to validate.

Accordingly, there is strong motivation to reflect on the purpose and scope of fuel cycle simulation in order to determine how one may investigate the suite of questions that interested parties have regarding the nuclear fuel cycle. The proposed work herein provides a methodological description of a generic simulation engine and interface that is agnostic to the different fuel cycle options that can be investigate. The interaction amongst facilities is abstracted away into a mathematical programming approximation of matching of supply and demand. Facilities themselves can behave in an automated or user-defined fashion, e.g., modules are currently in development to allow for on-the-fly isotopic calculations of spent reactor fuel based on run-time parameters. The facility building decision making is abstracted away into entities which own a given set of facilities. Finally, the demand for various facilities' services is abstracted away into entities modeled as regions. The various layers of abstraction from a simulation design point-of-view allow interested parties to investigate separate effects on their preferred fuel cycle metrics. Furthermore, the separation of the simulation engine from the various entities comprising the fuel cycle allows for easier and more straightforward testing and benchmarking.

Perhaps one of the strongest motivating factors of a renewed look at design and collaboration on nuclear fuel cycle simulation is the ability to foster and grow a simulation community. The proposed simulation framework is being implemented as CYCLUS [1], a nuclear fuel cycle simulator developed at the University of Wisconsin - Madison in conjunction with a number of other collabortors, including the University of Texas at Austin, the University of Utah, and the University of Idaho. CYCLUS is has an open-source codebase, allowing other interested developers to use or add to the existing simulation infrastructure. Critically, the fact that the source code is open allows for transparent investigation of the inner workings of the simulation. For example, other's efforts to benchmark their findings agaisnt a CYCLUS simulation are designed to be relatively easy in comparison to closed-source simulators, given the availability of the source code, input data, and output database.

BACKGROUND

Fuel Cycle Simulators

Previous implementations of fuel cycle simulators have varied both in methodology and distribution platform. The general purpose of all simulators is to model the flow of material among a variety of facilities in order to determine the viability of various proposed fuel cycles and their relative performance *vis-a-vis* a variety of metrics including resource utilization, costs, and proliferation resistance. However, there are a few key choices that have been historically made by all simulation developers,

including: what program or language to use, how to determine which facilities to build, when to build them, how to determine the flow of material when there are competing sources or sinks for that material, whether to model them individually, and at what level of fidelity physics should be modeled in the simulation (e.g., should material decay or should it not). One could describe these are the major design choices for simulation development teams to assess, and in general approaches are taken which span the gamut from the computationally "easy" to the computationally "complex" and the spectrum of almost full user-control to more substantial use of automated decision making.

The majority of fuel cycle simulation codes have been developed using a system dynamics platform. CAFCA [2], developed at MIT, was originally a MATLAB application, but is now written in the commercial software VENSIM [3]. VISION [4], developed by INL and originally based off of of the DYMOND code [5], has its simulation infrastructure written in the commercial software Powersim [6]. DANESS [7], originally developed at ANL and now moved to LISTO bvba in Belgium, is an iThink [8] application. Simulators that aren't written as commercial system dynamics applications tend to fall on either side of the spectrum of computational tools. COSI [9], for instance, is developed by the French CEA in Java. Cyclus [1], developed at UW-Madison, is a C++ application which provides an XML-input front end and SQL backend. Other fuel cycle simulations tend to be focused more on scoping calculations, for which spreadsheets can be applicable [10]. In general, all simulators (unless otherwise specified above) use Microsoft Excel to manage both input and output.

After choosing a development platform, the next basic simulation design decision deals with the modeling of facilities that act as nodes in a fuel-cycle material flow graph. There are, generally, two strategies that are taken by simulators: model aggregate fleets of facilities or model individual facilities. The former approach is taken by both CAFCA and VISION [11, 12]. Both of these system dynamics based simulators keep track of general fleet parameters as a function of time, e.g. the number of facilities under construction, in operation, or being decommissioned. Furthermore, they model the material flows into and out of the fleets; however, there is no distinction between individual facilities – the fleets themselves are modeled by the system dynamics equations. COSI, while not a system dynamics code, also models the reactors in their simulation as a fleet [13]. DANESS, though, takes a different tactic, tracking the history of each facility individually [14]. Adding this capability, for instance, allows DANESS to model individual reactors as unfueled or 'on hold'.

Fuel cycle simulator developers then must decide how building decisions are to be made in their models. For example, if both fast reactor and light water reactor technologies are available to the simulation at some time, how should it decide which to build, given that there is some demand? This decision is also necessary for supporting facilities, e.g. separations and fuel fabrication facilities, if they are modeled explicitly. Most simulators follow one of three paths: allow user-defined deployment, provide a heuristic that automates deployment, or combine the two. CAFCA, for instance, guarantees that reactors can be fueled for their entire lifetime, and thus builds reactors given a priority ordering, skipping candidates if a look-ahead heuristic determines there is not enough fuel [11]. In general, reactors in CAFCA are built to minimize light water reactor spent fuel (i.e., fast reactors are built if possible). VISION follows a similar path, incorporating user-defined reactor preferences before applying an automated heuristic based on fast reactor fuel availability [12]. Neither VISION nor CAFCA allows reactors to be built without guaranteed fuel availability. DANESS again takes a slightly different approach. It allows users to define reactor demand and will instantiate reactors based on a look-ahead heuristic that investigates whether a certain percentage of the reactor's required fuel is currently available. This is effectively a 'middle of the road' solution, that guarantees reactor fuel availability up to some user-defined percentage [15]. Accordingly, reactors that aren't fueled go 'on-hold', as previously discussed. It should be noted that each of the deployment decisions is based on a "class" of reactor (e.g. a PWR), and that each simulator described above has some limit on the number of "classes" of reactors. Importantly, each of these deployment methodologies are "hard-coded" into each simulator, i.e., they are the basis of the simulator framework and can not be changed in general.

The final two classes of simulation design decisions deal with the detail at which one wishes to model the actual materials in the simulation and at what level of detail to make material flow decisions based on those materials. Most simulators generally model physics at a basic level. For example, the decay of isotopes is available but not used in CAFCA simulations [2]. Material flow decisions are generally made based on aggregate quantities, e.g. kilograms of plutonium or transuranics [11]. VISION allows for decay of isotopics in its simulations, but does not generally make material flow decisions based on the updated isotopics, instead depending on user-defined input and output reactor fuel recipes [4]. COSI tends to take the most rigorous approach in this regard. It bases material flow decisions on equivalence methods, and models in-pile and decay calculations explicitly using a combination of depletion and transport codes, including CESAR, APOLLO, and ERANOS [16]. While COSI models isotopic transmutation and decay in order to inform their reactor fuel availability calculations, other simulators generally model isotopic changes to inform output metrics, such as repository capacity (which is a heavy function of input material radiotoxicity and decay heat).

Agent Based Supply Chain Simulation

Taking a step back from a review of available fuel cycle simulators, an investigation of the underlying structure of the supply-demand modeling paradigm shows that it is, in fact, a

subset of the family of supply-chain models. Further, if one wishes to model individual facilities, each of which can make independent decisions, one naturally arrives at the concept of agent-based modeling.There are a number of agent-based supply chain frameworks and implementations available in the literature with varying levels of accessibility due to proprietary considerations [17, 18, 19, 20]. However, the nuclear fuel cycle presents a few unique characteristics not explicitly treated in the literature. Perhaps the most difficult consideration we have identified is the need to specify target fuel recipes and match suppliers and consumers based on the requested recipe, i.e. there are both quantity and quality constraints placed on a requested commodity. An additional difficulty arises with the enforcement of regional-boundary constraints (e.g. prohibiting HEU trade between regions) and inter-enterprise preferences. Julka [18] discusses an extension to the normal supply-demand matching paradigm that allows for initial information to be provided to consumers before supply and demand are matched. A full discussion of this adaptation is provided in the methodology section.

Mathematical Programming

The general suite of techniques we propose to use to solve the supply-demand matching problem at each time step lie in the realm of mathematical programming. For general linear optimization problems (i.e., those without special structure), the Simplex Method, originally developed by George [21] in the 1950's, is a very well studied method for solving linear programs. It is computationally efficient, effectively solving the linear program (LP) by moving from feasible solution to feasible solution in a hill-climbing fashion, guaranteeing a increase in the optimality condition. An LP formulation of the nuclear fuel cycle is provided in the methodology section. It is foreseeable that such a formulation may not be adequate for all simulations because solutions can be provided in a fractional manner. While this is appropriate for certain fuel cycle services (e.g. using a fraction of an enrichment facility's SWU capacity), it does not necessarily accurately model reactor orders. Certainly, heuristics can be applied, but mathematical programming includes a suite of tools that can be used to solve problems with integer constraints. The analog to the Simplex Method (i.e., the most ubiquitous method used in practice) for optimization problems with integer decision variables is the Branch and Bound Method, first presented by Land and Doig [22]. It also has been used prolifically in the optimization community. A mixed integer-linear program (MILP) formulation of the nuclear fuel cycle is provided in the methodology section.

METHODOLOGY

This section lays out a plan by which a fully agent-based simulation can be implemented for a generic fuel cycle. The term "generic" implies that facilities involved are not known *a priori* and, accordingly, facilities can be coupled together as the designer wishes. For example, a designer has the choice to model a separations facility and advanced fuel fabrication facility as separate entities whose connected supply and demand are met by a generic engine, or to model the two facilities as a single combined and coupled entity. Additionally, the solution framework for this matching engine must be agnostic regarding the classes of materials involved. Rather than hard-coding in constraints and capacities for different material classes, they are added dynamically based on the entities involved in the solution. Included is a discussion of the proposal for entity interaction within the generic fuel cycle simulator framework. The goal of such a discussion is to identify the different design decisions made by the authors of the various simulators and to encapsulate the design decisions into the appropriate entities in the CYCLUS simulation. CYCLUS is designed to provide a minimal framework for agent-based simulation while maximizing the capability of future developers to adapt reusable sections of code to implement their own design decisions where appropriate.

Agent Interaction in CYCLUS

Deciding how a simulation is structured from an interactions standpoint is a delicate balance of known necessity and perceived future needs. There are basic decisions to make: do you want a system with discrete material transfers or continuous material flows? Discrete transfers more closely match reality and may provide insights in that regard, however the require more of their modeling apparatus due to messaging needs and other structures. More complex decisions include how one wants to determine connections between facilities. Do we assign supplier-consumer pairs to facilities? Do we allow them to change? Should the facility make such a decision? Should that decision be affected by any other entities? Tellingly, after being exposed to a number of different simulators through a benchmarking exercise, Guerin comments that the "operation of a fuel cycle model is as much art as science" [15]. These simulation-engine decisions comprise the art-related portion of fuel cycle simulation. The goal of any simulator is to make these decisions in as informed a way as possible according to domain-level knowledge, taking into account known and perceived requirements. In general, we attempt to minimize the sheer number of choices made in this regard, instead relying on well known and well documented practices of computer scientists and systems engineers.

CYCLUS has an additional goal in that we wish our core simulation infrastructure to be as flexible as possible. Given a few basic tenets of agent interaction, other developers should be able to create a new agent to "plug in" to the simulation. Accordingly, we must define a minimal set of behaviors to sufficiently inform the simulation infrastructure to run the simulation. This freedom allows us to run the simulation program and attach agents at run time, effectively separating the simu-

lation engine's functionality from the agents in the simulation. From an ecosystem point of view, being an open source code and having such capability allows expansion of the user and developer base into areas and institutions concerned with security and privacy. Furthermore, developers could participate both privately and publicly, e.g. adding general capability to the CYCLUS core that is needed for some functionality without specifying the internals. This open-source ecosystem further provides incentive to develop the agent-based simulation architecture. Other developers can concentrate their efforts on individual agent interaction, effectively encapsulating developer requirements for learning and interacting with the various simulation systems. For a more complete discussion on the ecosystem benefits of the CYCLUS development model, see [23].

Having decided upon agent-based interactions, one must determine a way to govern these interactions. We want to minimize agent dependency due to our above discussion, so using preference-based network flow formulations provide us with a viable solution technique that provides a consistent interface. The remainder of this section describes how that market resolution interface is informed by the agents. Basic agent simulation interaction, such as entering and leaving the simulation are also described.

Supply/Demand Parameters

The resolution of supply and demand at any given time step is the result of the mathematical program techniques discussed in the subsequent sections; however, there are simulation-engine details that must be described in order to set up that problem. The proposed resolution mechanism occurs in nominally three steps. The agent interactions include consumers and producers of a set of commodities and progresses is steps or "phases". The terminology of this "phase space" is taken from previous supply chain agent-based modeling work [18].

The first phase allows consumers of commodities to denote both the quantity of a commodity they need to consume as well as the target isotopics, or quality. Because the action is technically telling possible providers what type of product or service is required by a facility, this phase is termed the "Request for Bids Phase". This action is considered a "posting" of demand to the market exchange. It is possible that multiple commodities could meet a consumer's demand. Accordingly, consumers are allowed to "over-post", i.e., request a larger quantity of commodities than they can actually consume. The collection of commodities and quantities that make up a consumer's demand is termed a "demand portfolio", where each demanded commodity may also be accompanied by a target isotopic composition (termed its "quality" in CYCLUS parlance). A capacity constraint is required to accompany each portfolio, which facilitates the ability for consumers to "over-post". Additionally, if a portfolio is comprised of more than one commodity, the consumer must additionally provide a cardinal preference over the set of commodities. At the end of the posting phase, the market exchange will have a set of demand portfolios for each consumer.

For example, consider an LWR that can be filled with MOX or UOX. At a given time period in which it will order fuel, it would post a demand portfolio to the market exchange comprised of a quantity and quality of MOX and a quantity and quality of UOX. The demand portfolio must also indicate which fuel type it would prefer, i.e., the cardinal ordering mentioned above. Another example is that of an advanced fuel fabrication facility which fabricates fuel partially from separated material that has already passed through a reactor. Fuel assemblies produced by the facility are normally comprised of some prescribed quantity and quality of fissile material and the remaining volume is filled with fertile material, including depleted uranium from enrichment or reprocessed uranium from separations. Accordingly, a demand portfolio for that facility would be comprised of a quantity of depleted uranium, a quantity of reprocessed uranium, a total capacity, and a preference over the two.

The second phase allows suppliers to "respond" to the set of consumption portfolios. Each portfolio is comprised of requests for some set of commodities. Accordingly, for each request, suppliers of that commodity denote production capacities and an isotopic profile of the commodity they can provide. This isotopic profile is part of a heuristic mechanism to assign more fine-grained preferences among suppliers and consumers. Suppliers are allowed to offer the null set of isotopics as their profile, effectively providing no information. A supplier may have its production be constrained by more than one parameter. For example, a processing facility may have both a throughput constraint (i.e., it can only process material at a certain rate) and an inventory constraint (i.e., it can only hold some total material). Further, the facility could have a constraint on the quality of material to be processed, e.g., it may be able to handle a maximum radiotoxicity for any given time step which is a function of both the quantity of material is processes but also the isotopic content of that material. The formulation provided in the following section allows for multiple of such constraints as long as they are linear functions of the demanded commodity quantity. This phase is termed the "Bidding Phase" (analagous to Julka's Reply to Request for Quote phase[18]), and at its completion the possible connections between supplier and producer facilities, i.e., the arcs in the graph of the fuel cycle exchange, have been established with specific capacity constraints defined not only by the quantity of commodities that will traverse the arcs but also by the quality of the commodities.

The third phase of the supply-demand matching operation involves setting preference values for each supplier-consumer arc, termed the "Preference Assignment Phase". Supplier-consumer preferences are used to eventually set arc costs, which drive the solution of the variation of the multi-commodity transportation problem. Note that the cost translation technique itself is a simulation design decision. By this phase, each consumer has already assigned preferences as a function of commodity

amongst demands in its portfolio. The consumer is now allowed to inform the solver as to preferences amongst the responses to each request in each portfolio. From a facility point of view, the delineation between responses will nominally be made as a function of the quality of the material in the each response.

Take the reactor example provided above. Given an LWR request for MOX and UOX, multiple responses may be received to its request for MOX, each with different isotopic profiles of the MOX that can be provided. The reactor can then assign preference values over this set of potential MOX providers. Another example comes in the case of repositories. A repository may have a defined preference of material to accept based upon its heat load or radiotoxicity, both of which are functions of the quality, or isotopics, of a material. In certain simulators, limits on fuel entering a repository are imposed based upon the amount of time that has elapsed since the fuel has exited a reactor. It is in this phase that the CYCLUS engine would allow such capability. The time constraint is, in actuality, a constraint on heat load or radiotoxicity (one must let enough of the fission products decay). A repository could analyze possible input fuel isotopics and set the arc preference of any that violate a given rule to 0, effectively eliminating that arc. Once facilities have completed their preference assignments, their managers are able to permute them based on institutional or regional factors, as explained below.

Facilities

Facilities in CYCLUS are abstracted to either consumers or suppliers of commodities, and some may be both. Supplier agents are provided agency by being able to communicate to the market-resolution mechanism a variety of production capacity constraints in phase two of the solution setup. Consumer agents are provided agency by being able to assign preferences among possible suppliers based on the supplier's quality of product. Because this agency is encapsulated for each agent, it is possible to define strategies that can be attached or detached to the agents at run-time.

Institutions

Institutions in CYCLUS manage a set of facilities. They are tasked with the actual instantiation of specific facilities, for example, it is the institutions job to manage which facilities are commissioned and decommissioned at the appropriate times. The goal of including a notion of institutions is to allow an increased level of detail when investigating regional-specific scenarios. For example, there exist multi-national enterprises, such as AREVA, that operate fuel cycle facilities in a variety of countries, or regions. Furthermore, there are international governmental organizations, such as the IAEA, which plans to operate fuel cycle facilities that service other facilities in a variety of regions, e.g., a fuel bank. Accordingly, institutions in CYCLUS are able to augment the preferences of supplier-consumer pairs that have been established in order to simulate a mutual preference to trade material within an institution or based on institution-level relationships. Of course, situations arise in the real world where an institution has the capability to service its own facilities, but choose to use an outside provider because of either cost or time constraints, and such a situation is allowed in this framework as well. It is not immediately obvious to what degree institutions should be allowed to affect their managed facilities' preferences. Accordingly, through the course of implementation and experience modeling the fuel cycle with this framework, best practices will be determined.

Regions

Regions in CYCLUS are concerned with meeting certain requirements, e.g. power capacity, fuel cycle service capacity, etc. The notion of which facilities will meet this capacity is abstracted away from the region into the set of institutions that operate in that region. The amount of knowledge a region has regarding the types of facilities that can meet these external requirements and the extent to which that knowledge is used for facility deployment decisions depends on the region's implementation. For example, in the case of nuclear power capacity, any region knows that it needs additional reactors to be built. A "naive" implementation leaves the building of those reactors to the institutions that operate in the region. An "informed" implementation can order which reactors should be built by which institutions. The fundamental driver for such a design decision is a desire to abstract the management of facilities away from the decision of which facilities to build. It is important to note here that this abstraction allows for different deployment algorithms to be tested and exchanged in the CYCLUS framework without necessitating changes to the simulation engine, as is the case with other simulators described in previous sections and is consistent with the types of simulation design decisions required to maintain both flexibility and reusability.

Regions are further provided agency by their ability to affect preferences between supplier-consumer facility pairs in the third phase of the market resolution algorithm. The ability to perturb arc preferences between a given supplier and a given consumer allows fuel cycle simulation developers to model relatively complex interactions at a regional level. Examples of such interactions include the notion of tariffs – a region may prefer that facilities that can trade within its borders do so. Further, one could model the effect of sanctions – a region may not allow trade between facilities within its borders and another specified region. Because material quality information is also provided via the market resolution procedure, constraints can be applied on such information. For example, a region could scan the set of possible material leaving its borders via supplier transactions. It could then reset preferences of transactions that violate some decision rule, such as a restriction of plutonium-based fuels from exiting its boundaries. These principles can even exist on a spectrum that is a function of either quality (e.g., a limit could be placed on ^{239}Pu content) or region (e.g., a limit could be placed on plutonium-based fuels being provided

to another specific region).

The Generic Fuel Cycle Transportation Problem

Overview

Once a particular instance of a supply and demand commodity market has been described for a given time step, supplying facilities must somehow be matched with consumer facilities. It is possible to do this matching in a naive fashion using some heuristic, and such an approach may be appropriate for some simulations depending on the level of detail required. However, we seek a more flexible solution technique that will scale with the detail needed for each simulation. The mathematical programming discipline provides a natural fit for such solution techniques through a standard framework used to describe problem instances and a standard solution algorithm. Of the special problem structures enumerated in mathematical programming texts, the well-studied Transportation Problem provides a good starting point for the solution framework we seek.

The Transportation Problem is a member of the family of minimum-cost network flow problems. It is comprised of a set of suppliers, I, a set of consumers, J, and supplier-consumer costs, $c_{i,j}$. Suppliers and consumers form a bipartite graph, providing the problem with significant underlying structure. Its formulation is straightforward:

$$\min_z z = \sum_{i\in I}\sum_{j\in J} c_{i,j}x_{i,j} \tag{1a}$$

$$\text{s.t.} \sum_{j\in J} x_{i,j} \le s_i \qquad \forall\, i \in I \tag{1b}$$

$$\sum_{i\in I} x_{i,j} \ge d_j \qquad \forall\, j \in J \tag{1c}$$

$$x_{i,j} \ge 0 \qquad \forall\, x \in X \tag{1d}$$

where $x_{i,j}$ is the flow some commodity from i to j. Equation 1b provides a supply-side constraint, stating that the amount of material leaving a supplier, i, must not be greater than its supply, s_i, and Equation 1c provides a demand-side constraint, stating that the amount of material entering a consumer, j, must be at least the demanded amount, d_j. Equation 1d simply states that flow along arcs must be positive. Finally, the set X is the feasible solution space.

The Transportation Problem can be extended to include multiple commodities, resulting in the Multicommodity Transportation Problem (MTP):

$$\min_z z = \sum_{i\in I}\sum_{j\in J}\sum_{h\in H} c^h_{i,j}x^h_{i,j} \tag{2a}$$

$$\text{s.t.} \sum_{j\in J} x^h_{i,j} \le s^h_i \qquad \forall\, i \in I, \forall\, h \in H \tag{2b}$$

$$\sum_{i\in I} x^h_{i,j} \ge d^h_j \qquad \forall\, j \in J, \forall\, h \in H \tag{2c}$$

$$\sum_{h\in H} x^h_{i,j} \le r_{i,j} \qquad \forall\, i \in I, \forall\, j \in J \tag{2d}$$

$$x^h_{i,j} \ge 0 \qquad \forall\, x \in X \tag{2e}$$

where, conceptually, the main addition is Equation 2d, which is termed a "bundle" constraint, limiting the total flow of all commodities along an arc, (i, j), to some value, $r_{i,j}$. The MTP is perturbed by adding special considerations discussed below to formulate the Generic Fuel Cycle Transportation Problem (GFCTP).

Linear Program Formulation of the GFCTP

The linear program formulation of the GFCTP involves a set of players that act in an overarching market. The players are the supplier and consumer faciliteis involved in the simulation, i.e., the reactors, fabrication facilities, repositories, etc. In strict mathematical programming parlance, the GFCTP can be described as a Multicommodity Transportation Problem with Side Constraints. Accordingly are a number of departures from the classical MTP.

To begin, the multicommodity aspect of the problem is not manifest on arc capacities. Instead, facility demand constraints, whose equivalent in the MTP are Equation 2c, incorporate a set of satisfactory commodities, rather than a single satisfactory commodity. For example, a reactor may be able to accept UOX or MOX fuel, but has a demand for total fuel. Additionally, supplier facilities may have a set of constraints on their ability to supply a given commodity and they may not be able to directly express those constraints with the unit of the commodity market, which is generally kilograms of a commodity. Take an enrichment facility for example. Such a facility has nominally two constraints: SWU capacity and natural uranium capacity. The former constraint is a processing constraint, and the latter is an inventory constraint; however, both are necessary to fully define the problem. Furthermore, let us note that the output of this facility is kilograms of enriched uranium. Accordingly, the above capacities must be translated into units of this output. Because multiple commodities can satisfy demand, consumers denote a cost preference over the possible commodities they consume. Suppliers denote one or more production capacities for a given commodity which serve as the set of supply capacities analogous to the normal MCTP (Equation 2b). The GFCTP-LP formulation is as follows:

$$\min_{z} z = \sum_{h\in H}\sum_{i\in I}\sum_{j\in J} c_{i,j}^{h} x_{i,j}^{h} \tag{3a}$$

$$\text{s.t.} \sum_{j\in J} \beta_{i,k}(q_j^h) x_{i,j}^{h} \leq s_{i,k} \quad \forall k \in K_i^h, \forall i \in I, \forall h \in H \tag{3b}$$

$$\sum_{i\in I}\sum_{h\in H_j} x_{i,j}^{h} \geq d_j(H_j) \quad \forall j \in J \tag{3c}$$

$$x_{i,j}^{h} \geq 0 \quad \forall x \in X \tag{3d}$$

The sets and variables involved are described in Tables 1 and 2. Note that H_j is a subset of the commodities:

$$H_j \subseteq H \forall j \in J \tag{4}$$

Set	Description
H	all commodities
I	all producers
J	all consumers
X	the feasible set of flows between producers and consumers
K_i^h	the set of constraining capacities for producer i of commodity h
H_j	the set of satisfying commodities for consumer j

TABLE 1: Sets Appearing in the GFCTP-LP Formulation

Variable	Description
$c_{i,j}^h$	the unit cost of commodity h for producer i and consumer j
$x_{i,j}^h$	a decision variable, the flow of commodity h for producer i and consumer j
q_j^h	the requested quality of commodity h by consumer j
$\beta_{i,k}(q_j^h)$	a capacity translation function for capacity constraint k of producer i given q_j^h
$s_{i,k}^h$	a supply capacity of producer i corresponding to capacity constraint k of commodity h
$d_j(H_j)$	the total demand of consumer j over the set of satisfying commodities H_j

TABLE 2: Variables Appearing in the GFCTP-LP Formulation

This formulation deviates from the normal MCTP formulation via the expansion of capacity constraints (Equation 3b) and the inclusion of a constraint allowing multiple commodities that are able to meet the demand of a producer (Equation 3c). The former constraint maintains the multi-commodity nature of the formulation.

Under certain conditions, the GFCTP-LP will result in a simpler problem. The first possible condition is that each consumer could have its demand met by only one commodity, i.e.,

$$\left|H_j\right| = 1 \forall j \in J. \tag{5}$$

In such a situation, the GFCTP-LP can be transformed into an analog of the separable transportation problem as shown in [24]. Such a condition will effectively allow one to solve N different instances of a single-commodity problem, where N is the cardinality of H.

The second simplifying condition is if the constraining capacity set has a cardinality of unity, i.e.,

$$\left|K_i^h\right| = 1 \forall i \in I, \forall h \in H. \tag{6}$$

If both Equation 6 and 5 hold, then the GFCTP-LP is in fact the a normal Transportation Problem, because the quality translation function ($\beta_{i,k}(q_j^h)$) translates to a constant at solution time.

These simplifications are important to the computation time required to solve the resulting problem instance. The general solution technique for LPs is the Simplex Method, as previously described. Klee and Minty show that in the worst case, the Simplex Method will execute in exponential time [25], but in practice it is generally considered very computationally efficient. If the problem can be simplified to a TP, then the Transportation Simplex Method can be used [26].

Capacity Translation Function and Constraints Example The notion of a capacity translation function is something that has been introduced out of necessity due to the complexity of the GFCTP. Accordingly, an example will help clarify its purpose. This time can also be used to provide an example of a producer with multiple capacity constraints for a given commodity.

Take, for example, an enrichment facility. Such a facility produces the commodity "Enriched Uranium (EU)". This facility has two constraints on its operation for any given time period: the amount of Separative Work Units (SWU) that it can process, $s_{enr,SWU}$ and the total natural uranium (NU) feed it has on hand, $s_{enr,NU}$. Note that neither of these capacities are measure directly in the units of the commodity it produces, i.e., kilograms of enriched uranium (EU). The set of values for K_i^h for this facility are:

$$K_{enr}^{EU} = \{\text{SWU}, \text{NU}\} \tag{7}$$

Consider a set of requests for enriched uranium that this facility can possibly meet. Such requests have, in general, two parameters: P_j, the total product quantity (in kilograms), and ε_j, the product enrichment (in w/o U-235). The notation for enrichment, ε_j, is chosen over its normal form, x_p, to limit confusion with the LP notation of material flow, $x_{i,j}^h$. For the purposes of this constraint set, the quality of material in question is its enrichment, i.e.,

$$q_j^{EU} \equiv \varepsilon_j. \tag{8}$$

These values are set during a prior phase of the overall matching algorithm, and can therefore be considered constant. Further, let us note that, in general, an enrichment facility's operation, or rather its capacity, is governed by two parameters:

$\varepsilon_{f,enr}$, the fraction of U-235 in its feed material, and $\varepsilon_{t,enr}$, the fraction of U-235 in its tails material. These parameters determine the amount of SWU required to produce some amount of enriched uranium:

$$\begin{aligned} SWU = P(V(\varepsilon_j) + \frac{\varepsilon_j - \varepsilon_{f,enr}}{\varepsilon_{f,enr} - \varepsilon_{t,enr}} V(\varepsilon_{t,enr}) \\ - \frac{\varepsilon_j - \varepsilon_{t,enr}}{\varepsilon_{f,enr} - \varepsilon_{t,enr}} V(\varepsilon_{f,enr})) \end{aligned} \tag{9}$$

P in Equation 9 is the amount of produced enriched uranium, and $V(x)$ is the value function,

$$V(x) = (1-2x)\ln\left(\frac{1-x}{x}\right) \tag{10}$$

Utilizing the above equations, one can denote the functional forms of the arguments of this facility's two capacity constraints.

$$\beta_{enr,NU}(\varepsilon_j) = \frac{\varepsilon_j - \varepsilon_{t,enr}}{\varepsilon_{f,enr} - \varepsilon_{t,enr}} \tag{11}$$

$$\begin{aligned} \beta_{enr,SWU}(\varepsilon_j) = V(\varepsilon_j) \\ + \frac{\varepsilon_j - \varepsilon_{f,enr}}{\varepsilon_{f,enr} - \varepsilon_{t,enr}} V(\varepsilon_{t,enr}) \\ - \frac{\varepsilon_j - \varepsilon_{t,enr}}{\varepsilon_{f,enr} - \varepsilon_{t,enr}} V(\varepsilon_{f,enr}) \end{aligned} \tag{12}$$

These constraints correspond to the per-unit requirements for enriched uranium of natural uranium feed and SWU. Finally, we can form the set of constraint equations for the enrichment facility by combining Equations 3b, 8, 11, and 12.

$$\sum_{j \in J} \beta_{enr,NU}(\varepsilon_j)\, x^{EU}_{enr,j} \leq s_{enr,NU} \tag{13}$$

$$\sum_{j \in J} \beta_{enr,SWU}(\varepsilon_j)\, x^{EU}_{enr,j} \leq s_{enr,SWU} \tag{14}$$

Satisfying Commodity Set Example The other departure the GFCTP-LP takes from the normal MCTP formulation is the location of its multicommodity dependence. As presented above, the MCTP formulation includes a multicommodity arc capacity constraint, Equation 2d. There is no direct analog in the GFCTP, i.e., transportation arcs are assumed separate for separate commodities. There is still a notion of multicommodity dependence, however, via Equation 3c. This constraint models a situation in which different commodities can satisfy a consumer's demand.

Take the enrichment facility example, expanding on the previous discussion. Note that an enrichment facility takes feed uranium and then enriches its U-235 content. This feed uranium can come from different sources which have different feed enrichments. In practice, the most likely sources of feed uranium are natural uranium (NU) or recycled uranium (RU), a product of reprocessing light water reactor fuel. Recycled uranium may be advantageous to use if it has a higher weight percent of U-235 than does natural uranium. We can now state the set the values for H_j for this facility:

$$H_{enr} = \{\text{NU}, \text{RU}\} \tag{15}$$

Of course, the facility must define some preference function over the set of satisfying commodities. In this example, recycled uranium is more valuable because of its higher U-235 content, which translates into a (relatively large) SWU reduction in order to meet identical enrichment requests. This preference ordering is encapsulated in the cost function in Equation 3a. The nature of the cost function in the CYCLUS simulation environment is nontrivial and explained further in this section.

Mixed Integer-Linear Program Formulation of the GFCTP

The previous linear program (LP) formulation of the Generic Fuel Cycle Transportation Problem fully describes many of the types of transactions that arise at any given time step. However, it importantly glosses over the critical case of reactor fuel orders, which comprise a large amount of material orders within the simulation context. Specifically, it allows reactor fuel orders to be met by more than one supplier with an arbitrary amount of the order met by each supplier. Put another way, the LP formulation does not contain the discrete material information required to model the transaction of fuel assemblies. Such detail is not necessary in every simulation, but we wish to allow this advanced modeling for those that do need it. In order to provide this capability of quantizing orders, binary decision variables must be introduced and integer programming techniques must be utilized to solve the resulting mixed integer-linear program.

It should be noted that the addition of integer variables changes both the complexity of the formulation and the complexity of the solution technique. Such a change requires a Mixed Integer-Linear Program (MILP) formulation and solution via the branch-and-bound method which solves NP-Hard combinatorial optimization problems whereas the Linear Program (LP) version requires the simplex method which is much more efficient.

The updated formulation is presented below. The key difference is the inclusion binary variables $y^h_{i,j}$, which are 1 if producer i trades commodity h with consumer j and constants $\tilde{x}^h_j$, which denote the quantity of a quantized order. Further a new set is introduced, J_e, the set of consumers who require quantized, or exclusive, orders. The original set of consumers, i.e., those who allow partial orders, are denoted J_p. These two sets constitute the set of all consumers.

$$J = J_p \cup J_e \tag{16}$$

The Generic Fuel Cycle Transportation Problem with Exclusive Orders (GFCTP-E) formulation follows:

$$\min_{z} z = \sum_{h\in H}\sum_{i\in I}\sum_{j\in J_p} c^h_{i,j}x^h_{i,j} + \sum_{h\in H}\sum_{i\in I}\sum_{j\in J_e} c^h_{i,j}y^h_{i,j}\tilde{x}^h_j \tag{17a}$$

$$\text{s.t.} \sum_{j\in J_p} \beta_{i,k}(q^h_j)x^h_{i,j} + \sum_{j\in J_e} \beta_{i,k}(q^h_j)y^h_{i,j}\tilde{x}^h_j \leq s^h_{i,k}$$
$$\forall\, i\in I, \forall\, k\in K^h_i, \forall\, h\in H \tag{17b}$$

$$\sum_{i\in I}\sum_{h\in H_j} x^h_{i,j} \geq d_j(H_j) \qquad \forall\, j\in J_o \tag{17c}$$

$$\sum_{i\in I}\sum_{h\in H_j} y^h_{i,j}\tilde{x}^h_j \geq d_j(H_j) \qquad \forall\, j\in J_e \tag{17d}$$

$$\sum_{h\in H}\sum_{i\in I} y^h_{i,j} = 1 \qquad \forall\, j\in J_e \tag{17e}$$

$$x^h_{i,j} \geq 0 \qquad \forall\, x\in X \tag{17f}$$

$$y^h_{i,j} \in \{0,1\} \qquad \forall\, y\in Y \tag{17g}$$

The sets and variables involved are described in Tables 3 and 4. Note that H_j is a subset of the commodities:

$$H_j \subseteq H\, \forall\, j\in J_p, \forall\, j\in J_e \tag{18}$$

Set	Description
H	all commodities
I	all producers
J_p	all consumers who accept partial orders
J_e	all consumers who accept only exclusive orders
X	the feasible set of flows between producers and consumers
Y	the feasible set of exclusive flows between producers and consumers
K^h_i	the set of constraining capacities for producer i of commodity h
H_j	the set of satisfying commodities for consumer j

TABLE 3: Sets Appearing in the GFCTP-E Formulation

The examples of the various constraints from the previous section also apply here. The only difference is the notion of the binary variables, $y^h_{i,j}$, which denote a sort of on/off switch as to whether a consumer's entire requested amount of material is met by a supplier or not.

It should be noted that this advanced formulation adds significant complexity to the resolution method at every time step. However, simple heuristics exist. The most common of them is to solve a relaxed version of the problem in the form of a linear program, and to round values to form an integer solution. The exploration of additional heuristics will be performed based on the outcome of the implementation and analysis of this formulation in the CYCLUS simulation environment.

Variable	Description
$c^h_{i,j}$	the unit cost of commodity h for producer i and consumer j
$x^h_{i,j}$	a decision variable, the flow of commodity h for producer i and consumer j
q^h_j	the requested quality of commodity h by consumer j
$y^h_{i,j}$	a binary decision variable that is equal to 1 if there is flow from producer i to consumer j of commodity h
$\tilde{x}^h_j$	the amount of commodity h requested by consumer j
$\beta_{i,k}(q^h_j)$	a capacity translation function for capacity constraint k of producer i given q^h_j
$s^h_{i,k}$	a supply capacity of producer i corresponding to capacity constraint k of commodity h
$d_j(H_j)$	the total demand of consumer j over the set of satisfying commodities H_j

TABLE 4: Variables Appearing in the GFCTP-E Formulation

Cost Function

In any network flow problem, of which transportation problems are a subset, the cost of transporting commodities is what drives the solution. Accordingly, an accurate cost function is necessary to determine an accurate solution. Because the CYCLUS environment is still a nascent simulation platform, accurate pricing metrics, and what such metrics even are in terms of a centuries-long fuel cycle simulation, are generally difficult to ascertain, with the current standard source being the Advanced Fuel Cycle Cost Basis report [27]. Accordingly, the cost function is currently a measure of simulation entity preference, rather than a concrete representation of cost.

The notion of preference extends the work of Oliver's affinity metric [28]. The preference metric is generally consumer centric, i.e., consumers have a preference over the possible commodities that could meet their demand. For example, a reactor may be able to use UOX or MOX fuel, but may prefer to use MOX fuel. Such a preference differential allows the projection of real-world cost into the simulation. Additionally, the managers of a given facility, which in the CYCLUS simulation environment include its Institution and Region, also exert an influence over its preference. An obvious example is the concept of affinities given in [28]. In Oliver's work, an affinity or preference existed between facilities in "similar" institutions in order to drive the trading between institutions as a simple model of international relations. This idea is expanded upon to cover a facility's other managers and the commodities themselves. Additionally, a preference can be delineated between the proposed qualities of the same commodity from different vendors, e.g. if two vendors of MOX fuel exist. Finally, the notion of a preference is a positive one, and we require a notion of cost to solve the minimum-cost formulation of the multicommodity transportation problem with side constraints. Therefore

one must utilize a translation function.

Formally, we define a preference function, $\alpha_{i,j}(h)$, which is a cardinal preference ordering over a consumer's satisfying commodity set.

$$\alpha_{i,j}(h) \, \forall i \in I \, \forall h \in H_j \tag{19}$$

This ordering is a function both of the consumer, j, and producer, i. The dependence on producer encapsulates the relationship effects due to managerial preferences. We then define a cost translation function, f, that operates on the commodity preference function to produce an appropriate cost.

$$f : \alpha_{i,j}(h) \rightarrow c_{i,j}^{h} \tag{20}$$

A naive implementation, and perhaps all that is necessary for a proof-of-principle, is to define f as an inversion operator.

$$f(x) = \frac{1}{x} \tag{21}$$

The necessity for complexity of this translation function is not immediately obvious and an analysis will be performed to understand its impact.

RESULTS & CONCLUSIONS

This paper presented a generic methodology for modeling fuel cycle simulation entity interactions. The work was motivated by the desire to provide a flexible platform on which one could explore a variety of fuel cycles without requiring simulation-engine level changes in the underlying code base. Furthermore, this flexible platform is available as an open source project, allowing any developer access to the simulation engine while also allowing for cloistered development at sites with sensitive information.

Mathematical programming techniques were leveraged to allow such an encapsulation of simulation engine, abstracting away the notion of hard-coded connections between facilities. Two strategies were provided, one that runs at very quick time scales and can be adjusted via simple heuristics to model individual interactions using a linear programming formulation. A more advance, and also more computationally time consuming, approach was also provided for more detailed simulations, utilizing a mixed integer-linear programming approach.

Future work will concentrate first on implementing the proposed approach in the CYCLUS code base, building upon the already-existing agent-interaction simulation infrastructure. Concurrently, the market resolution algorithm will be benchmarked against other fuel cycle simulation codes in order to provide a basic level of confidence in it as well as to inform both the CYCLUS development group and the wider simulation community as to the relative strengths and weaknesses in a generic approach, rather than modeling specific fuel cycles explicitly. Additional work on the CYCLUS simulation engine will also continue, and will involve an incorporation of a graphical user interface front and back end, effectively removing the need to physically alter XML files or investigate SQL databases.

ACKNOWLEDGMENTS

This research is being performed using funding received from the DOE Office of Nuclear Energy's Nuclear Energy University Programs. The author thanks the NEUP for its generous support.

REFERENCES

1. P. WILSON, M. GIDDEN, K. HUFF, and R. CARLSEN, "Cyclus: A Nuclear Fuel Cycle Code from the University of Wisconsin Madison," (June 2012), http://cyclus.github.com/.
2. L. GUERIN and M. KAZIMI, "Impact of Alternative Nuclear Fuel Cycle Options on Infrastructure and Fuel Requirements, Actinide and Waste Inventories, and Economics," Technical Report MIT-NFC-TR-111, MIT Center for Advanced Nuclear Energy Systems (CANES), Cambridge, MA, United States (Sep. 2009).
3. P. VENSIM, "Ventana Systems, Inc," *Avaiable at: http://www. vensim. com* (2010).
4. J. JACOBSON, A. YACOUT, G. MATTHERN, S. PIET, D. SHROPSHIRE, R. JEFFERS, and T. SCHWEITZER, "Verifiable Fuel Cycle Simulation Model (VISION): A Tool for Analyzing Nuclear Fuel Cycle Futures," *Nuclear Technology*, **172**, 157–178 (Nov. 2010).
5. A. MOISSEYTSEV, "DYMOND, a Dynamic Model of Nuclear Development," *Argonne National Laboratory Internal Report* (2001).
6. P. STUDIO, "Powersim Software AS," (2003).
7. V. D. DURPEL, A. YACOUT, D. WADE, T. TAIWO, and U. LAUFERTS, "DANESS V4.2: Overview of Capabilities and Developments," in "Proceedings of Global 2009," Paris, France (Sep. 2009).
8. B. RICHMOND, S. PETERSON, K. CHICHAKLY, W. LIU, and J. WALLIS, "Ithink Software," *Isee Systems Inc, Lebanon NH* (2004).
9. L. BOUCHER and J. P. GROUILLER, ""COSI" : A Simulation Software for a Pool of Reactors and Fuel Cycle Plants," in "Fuel Cycle and High Level Waste Management," Beijing, China (May 2005).
10. M.-A. BRUDIEU, "Evaluation of Waste Streams associated with LWR Fuel Cycle Options – Focus on Steady State Recycling and Fabrication of PWR MOX and Recycled UOX fuel," (Jun. 2011).
11. R. BUSQUIM E SILVA, M. KAZIMI, and P. HEJZLAR, "A System Dynamics Study of the Nuclear Fuel Cycle

with Recycling: Options and Outcomes for the US and Brazil," Tech. Rep. MIT-NFC-TR-103, MIT Center for Advanced Nuclear Energy Systems (CANES), Cambridge, MA, United States (Nov. 2008).
12. T. M. SCHWEITZER, "Improved Building Methodology and Analysis of Delay Scenarios of Advanced Nuclear Fuel Cycles with the Verifiable Fuel Cycle Simulation Model (VISION)," (2008).
13. C. COQUELET-PASCAL and C. KIEFFER, "Validation of Physical Models Used in Scenarios Studies by Coupling COSI with ERANOS Package," in "Proceedings of Global 2011," Makuhari, Japan (Dec. 2011).
14. L. V. D. DURPEL, A. YACOUT, D. WADE, and H. KHALIL, "Daness dynamic analysis of nuclear system strategies," in "Global 2003: Atoms for Prosperity: Updating Eisenhouwer's Global Vision for Nuclear Energy," New Orleans, LA, United States (Nov. 2003).
15. L. GUERIN, L. VAN DEN DURPEL, B. DIXON, L. BOUCHER, and M. KAZIMI, "A Benchmark Study of Computer Codes for System Analysis of the Nuclear Fuel Cycle," Tech. Rep. MIT-NFC-TR-105, MIT (Apr. 2009).
16. M. MEYER and L. BOUCHER, "New Developments on COSI 6, the Simulation Software for Fuel Cycle Analysis," in "Proceedings of Global 2009," Paris, France (Sep. 2009).
17. J. SWAMINATHAN, S. SMITH, and N. SADEH, "Modeling Supply Chain Dynamics: A Multiagent Approach," *Decision Sciences*, **29**, 3, 607–632 (Jul. 1998).
18. N. JULKA, R. SRINIVASAN, and I. KARIMI, "Agent-based supply chain management-1: framework," *Computers & Chemical Engineering*, **26**, 12, 1755–1769 (2002).
19. D. J. VAN DER ZEE and J. VAN DER VORST, "A modeling framework for supply chain simulation: Opportunities for improved decision making," *Decision Sciences*, **36**, 1, 65–95 (2005).
20. D. C. CHATFIELD, J. C. HAYYA, and T. P. HARRISON, "A multi-formalism architecture for agent-based, order-centric supply chain simulation," *Simulation Modelling Practice and Theory*, **15**, 2, 153–174 (2007).
21. G. B. DANTZIG, A. ORDEN, and P. WOLFE, "The generalized simplex method for minimizing a linear form under linear inequality restraints," *Pacific Journal of Mathematics*, **5**, 2, 183–195 (1955).
22. A. H. LAND and A. G. DOIG, "An automatic method of solving discrete programming problems," *Econometrica: Journal of the Econometric Society*, pp. 497–520 (1960).
23. K. D. HUFF, P. P. WILSON, and M. J. GIDDEN, "Open Architecture and Modular Paradigm of Cyclus, a Fuel Cycle Simulation Code," in "Transactions of the American Nuclear Society," (2011), vol. 104, p. 183.
24. D. P. BERTSEKAS, *Network optimization: continuous and discrete models*, Athena Scientific Belmont, Massachusetts (1998).
25. V. KLEE and G. J. MINTY, "How good is the simplex algorithm," Tech. rep., DTIC Document (1970).
26. R. K. AHUJA, T. L. MAGNANTI, and J. B. ORLIN, "Network flows: theory, algorithms, and applications," (1993).
27. D. E. SHROPSHIRE, K. A. WILLIAMS, W. B. BOORE, J. D. SMITH, B. W. DIXON, M. DUNZIK-GOUGAR, R. D. ADAMS, and D. GOMBERT, "Advanced Fuel Cycle Cost Basis," Tech. rep. (Aug. 2009).
28. K. M. OLIVER, *GeniusV2: Software Design and Mathematical Formulations for Multi-Region Discrete Nuclear Fuel Cycle Simulation and Analysis*, Ph.D. thesis, University of Wisconsin-Madison (2009).