AIAA Guidance, Navigation and Control Conference and Exhibit
18 - 21 August 2008, Honolulu, Hawaii

AIAA 2008-7272

Optimal Seeding of a Class of Self-Reproducing Systems

Amor A. Menezes[*] and Pierre T. Kabamba[†]

University of Michigan, Ann Arbor, MI, 48109-2140, USA

This paper is motivated by the need to minimize the payload mass required to establish an extraterrestrial robotic colony. One approach for this minimization is to deploy a colony consisting of individual robots capable of self-reproducing. An important consideration for the establishment of such a colony is the identification of a seed. Since self-reproduction is achieved by the actions of a robot on available resources, a seed for the colony consists of a set of robots and a set of resources. Previous approaches to the seeding problem included certain restrictive assumptions about the self-reproducing system under consideration, possibly yielding seeds with an excessive number of resources. The algorithms were also incapable of seeding self-reproducing systems where the production of copies of progenitor robots required the assistance of their offspring. This paper outlines a novel algorithm to determine an *optimal* seed for a more general class of self-reproducing systems. Here, optimality is understood as the minimization of a cost function of the resources and robots, and the class of self-reproducing systems is more general because it allows the generation of a copy of a robot to depend on the assistance of its offspring. Illustrations of the algorithm are provided.

I. Introduction

Recent scientific research in self-reproduction has raised the prospect of advances in such diverse areas as space colonization, bioengineering, evolutionary software and autonomous manufacturing. Inspired by the work of John von Neumann,[1] extensive study of self-reproducing systems has taken place, including cellular automata, computer programs, kinematic machines, molecular machines, and robotic colonies. A comprehensive overview of the field is documented in Refs. 2 and 3.

Von Neumann postulated the existence of a threshold of complexity below which any attempt at self-reproduction was doomed to degeneracy. However, he did not define either complexity or degeneracy, nor did he compute the threshold's value. An extensive literature survey[4] indicates that no one had published an evaluation of this threshold in the following 60 years. Recently, Ref. 5 developed a novel theory of generation that is able to compute this von Neumann threshold. The results included a necessary and sufficient condition for non-degenerate offspring, i.e., offspring with the same reproductive capability as the progenitor. A probabilistic version of these results[6] also demonstrated parallels with information theory. Refs. 7 and 8 extended these results by providing algorithms that identified a seed for a restricted class of self-reproducing systems. This work builds on the previous papers by providing a seeding algorithm that is applicable to a more general class of self-reproducing systems. Additionally, it is demonstrated that the seed produced by this algorithm is optimal.

The remainder of this section presents a rationale for the identification of a seed for a self-reproducing system and discusses what makes the general seeding problem difficult. Section II highlights the results of prior solutions to seeding while examining their limitations. Section III details the necessary assumptions and definitions for seed identification, outlines a Seed Identification (SI) algorithm, and analyzes its properties. Section IV illustrates the application of the algorithm to self-replicating systems documented in the literature. Section V presents conclusions.

[*]Ph.D. Candidate, Department of Aerospace Engineering, François-Xavier Building, 1320 Beal Avenue.
[†]Professor, Department of Aerospace Engineering, François-Xavier Building, 1320 Beal Avenue.

A. Motivation

Within the context of extra-terrestrial colonization, current phased approaches to Martian exploration see the development of an enduring robotic presence on the Moon in the next five years. Several space agency roadmaps, of which Ref. 9 is typical, suggest that individual countries will deploy advanced robots as-needed to expand the size of an established colony. It is well known, however, that for every unit mass of payload to be launched into space, eighty additional units of mass are required to be launched as well[10] – hence the motivation to endow robots with the capacity for self-reproduction. These machines would be able to utilize on-site resources to enlarge their numbers when deemed necessary for a given task. Extra-terrestrial systems with such capability are less dependent than traditional colonies on the fiscal constraints of multiple launches of robots. Self-reproduction may therefore provide a highly cost-effective solution to the problem of establishing extra-terrestrial colonies.

To minimize mass, it would be even more efficient to identify the required elements for the initiation of a self-reproducing system, and send the smallest quantity of these elements into space. Although our previous papers on the subject identified this minimal seed, the proposed algorithms were unable to seed self-reproducing systems where the production of copies of progenitors required the assistance of their offspring. Such systems are prevalent; examples include the Krebs cycle in a cell,[11] the atmospheric ozone cycle and the atmospheric ozone cycle attacked by chlorine,[12] the nitrogen cycle when starting a new aquarium,[13] and some manufacturing systems.[14] The inclusion of this seeding capability and the identification of an optimal seed for such systems is the goal of this paper.

B. Background

The theoretical framework of this paper is Generation Theory,[5] which formalizes self-reproduction by "machines," a term describing any entity that is capable of producing an offspring regardless of its physical nature. A robot, a bacterium, or even a piece of software code is considered to be a machine in this theory if they can each produce another robot, bacterium or some lines of code respectively. These machines utilize resources to self-reproduce. A selected resource is manipulated by the parent machine via an embedded generation action to produce an outcome, which itself may or may not be a machine. Thus we can state the following:

Definition 1. A *generation system* is a quadruple $\Gamma = (U, M, R, G)$, where

- U is a *universal set* that contains machines, resources and outcomes of attempts at self-reproduction;
- $M \subseteq U$ is a *set of machines*;
- $R \subseteq U$ is a *set of resources* that can be utilized for self-reproduction; and,
- $G : M \times R \to U$ is a *generation function* that maps a machine and a resource into an outcome in the universal set.

It is possible that $M \cap R \neq \oslash$, and also $M \cup R \neq U$, as illustrated in Fig. 1. The former implies that machines can be resources, and the latter states that outcomes of attempts at generation may be neither machines nor resources.

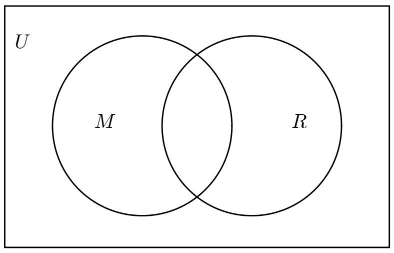

Figure 1. Pictorial representation of Definition 1.

When a machine $x \in M$ processes a resource $r \in R$ to generate an outcome $y \in U$, we write:

$$y = G(x, r). \tag{1}$$

In (1), we say that "x is capable of generating y," and we call the process *reproduction.* If we have $x = G(x, r)$ then we say that "x is capable of generating itself," and we call the process *replication.*

We also make use of concepts from graph theory[15] in this paper. Equation (1) may be represented by a *directed reproduction graph*, γ, as shown in Fig. 2. In this diagram, machine x and outcome y are vertices, resource r is an edge, and the direction of the edge indicates that it is machine x that uses resource r to generate outcome y.

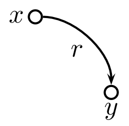

Figure 2. The directed reproduction graph of (1).

Definition 2. The *directed graph representation* of a generation system $\Gamma = (U, M, R, G)$ is the directed supergraph (V, E) containing all directed reproduction graphs that produce machines in M. Thus the vertex set, V, of the supergraph is equal to the machine set, M, and the edge set, E, of the supergraph is the set $\{r \in R | \ \exists \ x, y \in M : y = G(x, r)\}$.

When depicting a generation system, we only concern ourselves with machines that are produced rather than all possible generation outcomes. Hence, the above definition excludes outcomes of attempts at self-reproduction that are not machines.

We can use a compact notation to denote a sequential selection of resources. Let $(r_\mu) = (r_1, r_2, \ldots, r_\mu)$ be a sequence of μ resources from R. We define the notation

$$G(x, (r_\mu)) := G(\ldots G(G(x, r_1), r_2) \ldots, r_\mu)$$

to denote the outcome of generation using the sequence (r_μ) – see Fig. 3.

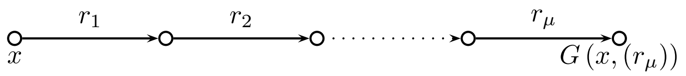

Figure 3. The directed graph representation of $G(x, (r_\mu))$.

We formally define a seed in Section III. Intuitively, since self-reproduction is achieved by the actions of a machine on available resources, a seed for a self-reproducing system consists of a set of machines and a set of resources such that all of the machines in the generation system are produced from a seed machine processing a finite sequence of seed resources.

C. Difficulty of the Seeding Problem

Many factors contribute to the inherent difficulty of seeding, including:

(a) the possibility that a given generation system is made up of multiple, disjoint subsystems, each with a different seed. Alternatively, there could be multiple, intersecting subsystems, with some common seed elements in each subsystem. Any seeding algorithm would have to be able to deal with both possibilities without any *a priori* knowledge about the system.

(b) the potential for generation cycles (sequences of generations resulting in the production of a machine identical to itself after n generations) within a given self-reproducing system. If these cycles exist, then one naturally wonders which of the machines in a particular cycle, if any, should belong to the seed.

(c) the fact that degenerate machines (machines whose progeny can no longer be machines) should not belong to the seed for a self-reproducing system. On the other hand, if all the machines in the generation system are degenerate, then there is a need to identify a machine that is of highest generation rank[5] to seed the system.

(d) the complexity of the resource set. A consistent theme in the literature is that a machine operates on an ordered list of elements constituting a resource. This list can include duplicates of elements contained in another resource that is also an ordered list. In addition, each list can also include machines.

(e) the existence of self-reproducing systems where the generation of a copy of a machine depends on the assistance of its offspring. Typically, this phenomenon manifests itself as a combination of (b) and (d), when a resource employed at some stage of a generation cycle contains a machine that is generated at a different stage in the cycle. It is not always clear whether to take the progenitor, its offspring, neither one, or both, to belong to the seed.

It is perhaps because of all of these factors that the seeding problem is still mostly open. The only known work in this area is our previous attempts at tackling restricted versions of this problem. Reference 7 resulted in the Seed Identification and Generation Analysis (SIGA) algorithm, and Ref. 8 presented a Restricted Seed Identification (RSI) algorithm that is applicable to a larger class of self-reproducing systems.

II. Prior Solutions to Seeding

A. The SIGA Algorithm

In Ref. 7, we allowed each resource to contain an ordered list of physical elements that could include machines. We therefore defined a containment relation as follows.

Definition 3. If machine x_i belongs to an ordered list of the elements of resource r_j, then we say that x_i is *contained* in r_j, and we write $x_i \prec r_j$, where "$\prec$" is the *containment relation*.

We assumed that if machine x was contained in resource r, $x \prec r$, then the ordered sub-list of the elements of r that did not contain the machine x also belonged to the set of resources, i.e., $r \backslash x \in R$. We also assumed that there existed a machine in the generation system that was capable of producing any machine in the system after μ generations. The idea of the algorithm was to remove all degenerate machines from the sets M and R, select one of the remaining non-degenerate machines to be a seed machine, and select the set $R \backslash M$ to be the resource seed set. In short, of the difficulties listed in Section I C, (c) was effectively handled, (b) and (d) were ineffectively handled, and (a) and (e) were not handled.

B. The RSI Algorithm

Reference 8 took a more general approach to the seeding problem, also presenting necessary and sufficient conditions to find an optimal seed for a larger class of generation systems. The paper developed an algorithm based on the following definition.

Definition 4. The generation system $\Gamma = (U, M, R, G)$ is *strongly regular* if, whenever $y = G(x, (r_{\mu_1}))$, where x and y are machines and (r_{μ_1}) is a sequence of μ_1 resources, we have $y \not\prec r$ for all $r \in (r_{\mu_1})$.

Thus, in a strongly regular generation system, if a machine is contained in a resource, then that resource cannot be utilized in any sequence of resources used to generate the machine. The idea of the RSI algorithm was to separately seed specific subsets of the generation system. Of the difficulties listed in Section I C, (b) and (c) were effectively handled, (a) and (d) were ineffectively handled, and (e) was not handled.

III. Seed Identification

This section formulates the Seed Identification Problem and presents an extended version of the RSI algorithm, the SI algorithm. We start by listing the basic assumptions required to seed a generation system.

Assumption 1. Both the number of machines and the number of resources are finite.

Assumption 2. An inexhaustible supply of each resource is available.

Assumption 3. All the machines in the generation system must be produced, although they need not all belong to a seed.

Assumption 4. If machine x is contained in resource r, $x \prec r$, then the ordered list of the elements of r that does not contain the machine x also belongs to the set of resources, i.e., $r \backslash x \in R$.

We define a seed as follows.

Definition 5. Let $\Gamma = (U, M, R, G)$ be a generation system. A *seed of order $\nu\mu$* for Γ is a set

$$\begin{aligned} S &= M_S \cup R_S, \text{ where} \\ M_S &= \{x_1, x_2, \ldots, x_\nu\},\ M_S \subseteq M, \text{ and} \\ R_S &= \{r_1, r_2, \ldots, r_\mu\},\ R_S \subseteq R, \end{aligned}$$

such that $\forall y_1 \in M, \exists \mu_1 < \infty, \exists (r_{\mu_1}) \in R_S, \exists y_0 \in M_S$, such that $G(y_0, (r_{\mu_1})) = y_1$.

That is, a seed for a generation system consists of ν machines and μ resources such that all of the machines in the generation system can be produced from a seed machine processing a finite sequence of seed resources.

We now relax the notion of strong regularity.

Definition 6. The generation system $\Gamma = (U, M, R, G)$ is *weakly regular* if whenever $y = G(x, r)$, where x and y are machines and r is a resource, we have $y \not\prec r$.

Thus, in a weakly regular generation system, no machine can be contained in any resource used to produce that machine. The difference between strong regularity and weak regularity lies in the location and number of resources where offspring containment is allowed. In a strongly regular generation system, containment of an offspring machine is not permitted in any sequence of resources used to generate that machine. In a weakly regular generation system, containment of an offspring machine is permitted in any sequence of resources used to generate that machine as long as the containment does not occur with the last resource of the sequence. It is important to note this distinction and identify how to seed such self-reproducing systems because we find weakly regular generation systems in nature.[11–13]

Lemma 1. *Strong regularity is a sufficient condition for weak regularity.*

Proof. See Appendix. □

Remark 1. The converse is not true.

Definition 7. A *family* is a generation system $\Gamma = (U, M, R, G)$ where $\forall (x, y) \in M, \exists z \in M$, and $(r_n), (r_m) \in R, n, m \geq 0$, such that $x = G(z, (r_n))$ and $y = G(z, (r_m))$. A *matriarch* of a family is an element $x_♀ \in M$ such that $\forall x \in M, x \neq x_♀, \exists (r_{\mu_1})$ selected from R such that $G(x_♀, (r_{\mu_1})) = x$.

Theorem 1. *Every family has a matriarch.*

Proof. See Appendix. □

Theorem 2. *The directed graph representation of a family is weakly connected.*

Proof. See Appendix. □

Assumption 5. We assume that the generation system to be seeded, $\Gamma = (U, M, R, G)$, is weakly regular and made up of one or more disjoint families.

Assumption 6. The function $J : M \to \mathbb{R}$ representing the cost of machines, and the function $K : R \to \mathbb{R}$ representing the cost of resources, are both provided.

We can now state the following.

Seed Identification Problem: minimize the total cost of the machines and resources in a seed under Assumptions 1 through 6.

The idea of the SI algorithm is that under Assumption 5, seeding the whole generation system may be accomplished by seeding each individual family. To seed by family, we need to determine all the descendants of a particular machine. This is facilitated by the notion of a "generation subsystem of a machine," which is a subset of a particular family and is itself a family.

Definition 8. The *generation subsystem of machine* x_1 is the generation system $\Gamma_{x_1} = (U, M_{x_1}, R_{x_1}, G)$ where,

$$\begin{aligned}
M_{x_1} &= \bigcup_{i=0}^{\infty} M_{x_1}^i \\
M_{x_1}^i &= \{x \in M | \ \exists (r_i) \in R : x = G(x_1, (r_i))\} \\
R_{x_1} &= \bigcup_{i=0}^{\infty} R_{x_1}^i \\
R_{x_1}^i &= \bigcup_{|R|^i} \{(r_i) \in R | \ G(x_1, (r_i)) \in M\}.
\end{aligned}$$

In Definition 8, $M_{x_1}^i$ is the set of all the descendants of x_1 produced after i generations, M_{x_1} is the set of all the descendants of x_1, $R_{x_1}^i$ is the set of all resource sequences of length i that would produce a descendant of x_1, and R_{x_1} is the set of all resource sequences that would produce a descendant of x_1. Hence, the generation subsystem of x_1 is the largest family for which x_1 is a matriarch.

We determine the subsystems for which there exists one machine capable of generating all other machines in the subsystem. It is among these subsystems that one may find a matriarch of a family. Consequently, individually seeding each of these subsystems of matriarchs seeds the whole family. Let $M_{♀}$ denote the set of matriarchs.

In the generation subsystem of a matriarch $x_{♀}$, every machine in the subsystem can be produced except possibly $x_{♀}$ itself. Thus, in the course of seeding the subsystem of $x_{♀}$, the machines to pick for the seed set of the subsystem, $S_{x_{♀}}$, are $x_{♀}$ and certain machines contained in $R_{x_{♀}}$.

The next theorem suggests an approach to seeding weakly regular generation systems, and replaces the sufficient condition for minimizing $|M_S|$ that was used by the RSI algorithm.

Theorem 3. *Assume that the generation subsystem of machine x_1, $\Gamma_{x_1} = (U, M_{x_1}, R_{x_1}, G)$, is weakly regular. Let $y \in M_{x_1}$, and $(r_{m+1}) \in R_{x_1}$. Suppose that:*

(1) $\forall i : 1 \le i \le m$, $x_{i+1} = G(x_1, (r_i)) \ne y$.

(2) $G(x_{m+1}, r_{m+1}) = G(x_1, (r_{m+1})) = y$.

(3) $y \prec (r_m)$.

Then a seed set for the weakly regular family $(U, (x_{m+1}),\ (r_m), G)$, is $S = \{x_1, y\} \cup \{(r_m) \backslash y\}$, and S has minimum $|M_S|$.

Proof. See Appendix. □

Theorem 3 states that if a sequence of m resources, (r_m), is used to produce a sequence of $m+1$ machines, (x_{m+1}), and machine y is contained in (r_m) but does not belong to (x_{m+1}), and the resource seed set is devoid of machines, then the machine seed set must consist of y and the first machine in (x_{m+1}). Hence, we must examine the sequences of machines generated by a matriarch when seeding weakly regular generation systems. We first present a Line Seeding (LS) sub-algorithm, before giving the general SI algorithm.

A. The LS Sub-Algorithm

Input: a simple path in the directed minimum spanning tree (DMST) representation of the weakly regular generation subsystem of a matriarch, x_1, that begins at x_1. The machines (vertices) in this path constitute M, and the resources (edges) used in this path constitute R. Let the path length be n.
Output: a seed set, S, for this simple path.
/*We create a new line graph, known as the "seeding graph," in order to help determine the output.*/
Start.

```
1: $M_S \leftarrow \{x_1\}$.
2: Initialize the seeding graph with one vertex, $x_1$, and zero edges.
3: for $1 \le i \le n$ do
```

```
4:     Let $y \leftarrow G(x_1, (r_i))$.
5:     if $y$ is not a vertex in the seeding graph then
6:         Add the machines contained in $(r_i)\backslash(x_1, y)$ that are not already on the seeding graph as new vertices.
           Draw a directed edge from the last vertex to the first contained machine, from the first contained
           machine to the second contained machine, and so on.
7:         Add $y$ to the end of this line graph with a directed edge that comes from the last contained machine
           that was added.
8:     else
9:         if $(x_i)\backslash x_1$ and all the contained machines in $(r_i)\backslash(x_1, y)$ are not between the $x_1$ and $y$ vertices on
           the line graph (any contained machines that are not already on the seeding graph may simply be
           added in an appropriate position) then
10:            $M_S \leftarrow M_S \cup \{y\}$
11:        end if
12:    end if
13: end for
14: $R_S \leftarrow R\backslash M$.
15: $S \leftarrow M_S \cup R_S$.
Stop.
```

At each iteration of the LS sub-algorithm, the number of intermediate machines grows by one. If a path is not strongly regular because of one of the intermediate machines, then Theorem 3 comes into play. The seeding graph created by the LS sub-algorithm is a tool to indicate which machines should be added to the machine seed set.

Theorem 4. *The LS sub-algorithm is correct. That is, the output of the LS sub-algorithm is a seed for a simple path in the DMST of the generation subsystem of a matriarch that starts at the root of the tree.*

Proof. See Appendix. □

B. The SI Algorithm

Input: a generation system of n machines and m resources that is weakly regular and made up of one or more disjoint families, and cost functions $J : M \rightarrow \mathbb{R}$ and $K : R \rightarrow \mathbb{R}$.
Output: a seed set, S, for this generation system.
Start.

```
1: for all $x_i \in M$, $1 \leq i \leq n$ do
2:     Determine $\Gamma_{x_i}$.
3: end for
   /*In the above, with each machine (vertex) as a starting point (root) in the initial generation system
   (directed graph), we need to find the subsystem (maximally connected subgraph) that can be generated
   (reached from the root). Two well known algorithms to compute the reachable components in a graph
   are the Breadth-First Search (BFS) and the Depth-First Search (DFS) algorithms.[16–18]*/
4: Select the $\Gamma_{x_i}$ where $|M_{x_i}| \geq |M_{x_j}|$, $\forall 1 \leq j \leq n$.
   /*The idea is to seed a primary subsystem first, and then go back to a secondary subsystem $M\backslash M_{x_i}$
   and partition and seed iteratively.*/
5: for all the matriarchs of the largest generation subsystem do
6:     if in the graph representation of $\Gamma_{x_i}$, $x_i$ has entering edges then
7:         Add a new vertex $x'_i$.
8:         Change these edges so that they now enter $x'_i$.
9:     end if
10:    Find the directed minimum spanning tree (DMST) in the graph of $\Gamma_{x_i}$ with root at $x_i$.
11:    $\Gamma_{x_i min} \leftarrow$ the DMST of $\Gamma_{x_i}$.
12:    for all simple paths in the DMST that begin at $x_i$ do
13:        Use the LS sub-algorithm to seed each path.
14:    end for
15:    $S_{x_i} \leftarrow \bigcup S_{path}$.
16: end for
```

```
17: Select the $S_{x_i}$ for which $\sum_{y \in M_{S_{x_i}}} J(y) + \sum_{r \in R_{S_{x_i}}} K(r)$ is a minimum.
    /*We ensure that the primary subsystem has the property that each offspring is generated from only
    one resource. Thereafter, we can select all resources to be a part of the seed set. We use the Chu-Liu-
    Edmonds algorithm[19,20] to find the DMST for each matriarch in the primary subsystem. For each of
    these DMSTs, we can apply the DFS algorithm to find all the simple paths that begin at the root and
    utilize the LS sub-algorithm to seed each path. The seed for the entire subsystem for a particular DMST
    is the union of the seeds for each path. We pick the DMST seed with minimum total cost.*/
18: Remove all $x \in M_{x_i}$ from $M$.
19: if $M \neq \oslash$ then
20:     Go to Line 4.
21: else
22:     $S \leftarrow \bigcup_{x_i \in M_{♀}} S_{x_i}$.
23: end if
Stop.
```

C. Properties of the SI Algorithm

Theorem 5. *The SI algorithm is correct. That is, the output of the algorithm is an optimal seed for the given generation system.*

Proof. See Appendix. □

Remark 2. The assumption of disjoint families, Assumption 5, is required to ensure optimality of the seed resource set. Although the proposed algorithm will work for families that are not disjoint, no claims about optimality can be made. However, we conjecture that the resultant seed will be close to optimal. If the system does not possess disjoint families, then Line 20 of the SI algorithm should read "Go to Line 1."

Theorem 6. *The SI algorithm is complete. That is, the algorithm will output a seed if one exists for the given generation system.*

Proof. See Appendix. □

Theorem 7. *The SI algorithm is guaranteed to stop after a finite number of iterations. The time complexity for the operation of this algorithm is polynomial in $|M|$ and $|R|$.*

Proof. See Appendix. □

IV. Example Applications of the SI Algorithm

Two examples are provided in this section. The first example serves to illustrate the broad workings of the SI algorithm by rendering the LS sub-algorithm trivial. The second example illustrates the details of the LS sub-algorithm and its relationship with the SI algorithm.

A. A Strongly Regular Self-Replicating System

We can use Generation Theory and the algorithm in this paper to analyze a modified version of the *Semi-Autonomous Replicating System* designed by Chirikjian et al.[21,22]

In the original design, we can take M to be the set of all entities that are each made up of two or more LEGO Mindstorm kit components fixed together in some way. Let

$$\begin{aligned} M &:= \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\}, \text{ and} \\ R &:= \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9\}, \end{aligned}$$

where we define each of the constituent machines and resources in the manner that follows. The sequence of generation steps is also outlined. The replication process is illustrated in Fig. 4.

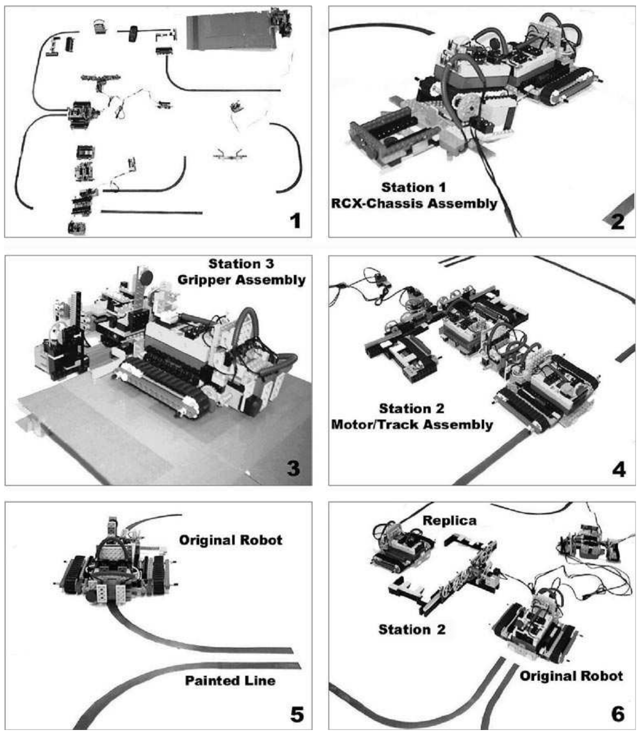

Figure 4. The semi-autonomous replication process of the Suthakorn-Kwon-Chirikjian robot.[21]

x_1 := prototype robot

r_1 := (conveyor-belt/sensor unit, docking unit, electrical connector, central controller unit (CCU), electrical cable)

x_2 := chassis assembly station

$x_2 = G(x_1, r_1)$

r_2 := chassis

x_3 := chassis aligned in assembly position

$x_3 = G(x_1, r_2)$

r_3 := (robot control system, x_3)

x_4 := RCX-chassis assembly

$x_4 = G(x_2, r_3)$

r_4 := gripper assembly/disassembly station := (CCU, electrical connector, ramp and lift system, gripper)

x_5 := prototype robot with gripper

$x_5 = G(x_1, r_4)$

$x_1 = G(x_5, r_4)$

r_5 := (left LEGO hook, right LEGO hook, CCU, electrical connector, stationary docking sensor, motorized pulley unit)

x_6 := motor and track assembly station

$x_6 = G(x_5, r_5)$

r_6 := (left LEGO track, right LEGO track)

x_7 := tracks aligned onto hooks

$x_7 = G(x_1, r_6)$

r_7 := (motor/sensor unit, x_4)

x_8 := RCX-chassis-motor assembly, moved to position

$x_8 = G(x_1, r_7)$

r_8 := (x_7, x_8)

x_9 := prototype robot on hooks

$x_9 = G(x_6, r_8)$

r_9 := x_9

$x_1 = G(x_1, r_9)$

The original Suthakorn-Kwon-Chirikjian generation system is a single family, and so the SI algorithm terminates after one iteration. If we take into account the necessity of batteries for operation, the application becomes non-trivial. We stipulate that the robot controller (RCX) runs on charged batteries, and that there is a battery charger running on a supply of readily available electricity that can charge uncharged batteries. Thus, the following changes to the system need to be made.

$$\begin{aligned} M &:= \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}, x_{11}\} \\ R &:= \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9, r_{10}\}, \end{aligned}$$

x_{10} := battery charger
r_{10} := (electricity, uncharged batteries)
x_{11} := charged batteries
$x_{11} = G(x_{10}, r_{10})$
and the definition for r_3 becomes
r_3 := (robot control system, x_3, x_{11})
It follows that we have the generation diagram indicated in Fig. 5.

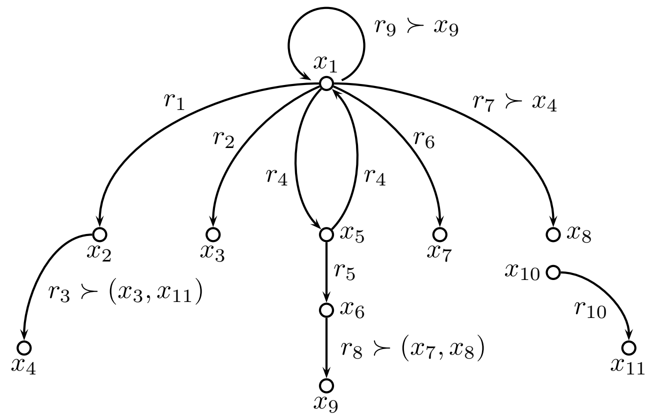

Figure 5. Directed graph representation of the modified Suthakorn-Kwon-Chirikjian semi-autonomous replicating system.

This generation system is strongly regular, and is made up of two disjoint families corresponding to the two disjoint subgraphs of Fig. 5. Applying the SI algorithm to this generation system yields an optimal seed for the system. To demonstrate the workings of the algorithm, we give a part of the output. For this example, we let $\sum_{y \in M_{S_{x_i}}} J(y) := |M_{S_{x_i}}|$, and $\forall r \in R$, $K(r) := |r|$.

The machine sets of $\Gamma_{x_i}, 1 \leq i \leq 11$ are the following.
$M_{x_1} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$
$M_{x_2} = \{x_4\}$
$M_{x_3} = \oslash$
$M_{x_4} = \oslash$
$M_{x_5} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$
$M_{x_6} = \{x_9\}$
$M_{x_7} = \oslash$
$M_{x_8} = \oslash$
$M_{x_9} = \oslash$
$M_{x_{10}} = \{x_{11}\}$
$M_{x_{11}} = \oslash$.
We can select either x_1 or x_5. Since the sets of machines that can be generated are equal, x_1 and x_5 must be matriarchs for the same family.

Consider x_1. Since x_1 has entering edges in the Fig. 5, we define a new vertex x_1' and change these edges so that they now enter x_1'. The DMST with root at x_1 yields $R_{x_1 min} = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9\}$. The only choice made by the DMST algorithm is the selection of r_9 over r_4 in generating x_1', since the former has lower cost. The LS sub-algorithm returns $S_{x_1} = \{x_1\} \cup R_{x_1 min} \backslash M$.

Similarly, for x_5, we define a new vertex x_5' and change the edges that enter x_5 so that they now enter x_5'. The DMST with root at x_5 yields $R_{x_5 min} = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}$. The DMST algorithm does not select r_9. The LS sub-algorithm returns $S_{x_5} = \{x_5\} \cup R_{x_5 min} \backslash M$.

Of the two matriarchs, S_{x_5} is selected since it has lower total cost.

Thus, the seed for the first family is

$$S_{x_5} = \{x_5\} \cup \{r_1, r_2, r_3 \backslash (x_3, x_{11}), r_4, r_5, r_6, r_7 \backslash x_4, r_8 \backslash (x_7, x_8)\}.$$

Continuing the SI algorithm, we remove the machines that are in the generation subsystem of x_5, leaving us with $M = \{x_{10}, x_{11}\}$.

We now iterate through the algorithm, selecting x_{10} since it has the larger generation subsystem. The DMST yields $R_{x_{10} min} = \{r_{10}\}$, and the LS sub-algorithm gives us one possible seed.

Thus, the seed for the second family is

$$S_{x_{10}} = \{x_{10}\} \cup \{r_{10}\}.$$

Removing the machines in the generation subsystem of x_{10}, leaves us with $M = \oslash$.

Therefore, a seed for the self-replicating system is

$$S = \{x_5, x_{10}\} \cup \{r_1, r_2, r_3 \backslash (x_3, x_{11}), r_4, r_5, r_6, r_7 \backslash x_4, r_8 \backslash (x_7, x_8), r_{10}\}.$$

We have thus arrived at a very logical, yet informative result – the battery charger and the prototype robot with the gripper can initiate the semi-autonomous replicating system. Contrary to intuition, the optimal seed does not include the prototype robot together with resources, but instead includes the prototype robot with gripper and fewer required resources.

B. A Weakly Regular Self-Replicating System

The example in this section serves only to illustrate the working of lines 5 through 17 of the SI algorithm. To demonstrate the applicability to *any* self-reproducing system, not just robotic ones, we use the naturally occurring atmospheric ozone cycle attacked by chlorine.[12] The generation diagram of this self-reproducing system is drawn in Fig. 6, using the model:

$$\begin{aligned} M &:= \{x_1, x_2, x_3, x_4, x_5\}, \text{ and} \\ R &:= \{r_1, r_2, r_3, r_4, r_5\}, \end{aligned}$$

where we define each of the constituent machines, resources and generation steps as follows.

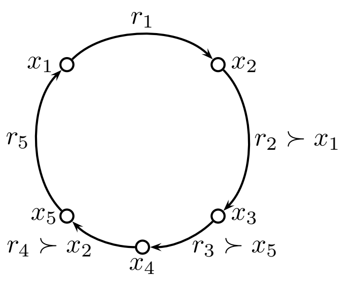

Figure 6. Directed graph representation of the atmospheric ozone cycle attacked by chlorine.

$x_1 := O_2$, or oxygen molecules
$r_1 :=$ ultraviolet radiation
$x_2 := O$, or excited oxygen atoms
$x_2 = G(x_1, r_1)$
$r_2 := (x_1$,neutral particle)

$x_3 := O_3$, or ozone molecules
$x_3 = G(x_2, r_2)$
$x_4 := ClO + O_2$
$x_5 := Cl + O_2$
$r_3 := x_5$ (note that just Cl is required)
$x_4 = G(x_3, r_3)$
$r_4 := x_2$
$x_5 = G(x_4, r_4)$
$r_5 := ()$, a dummy resource
$x_1 = G(x_5, r_5)$

This self-replicating system is made up of a single family, as indicated by the connected graph of Fig. 6. The family is not strongly regular; for instance, starting with machine x_3, we would need x_5 before we have produced it. However, the family is weakly regular; no offspring machine is contained in a resource used to generate that machine.

Every machine in this cycle is a matriarch for the family, and so the LS sub-algorithm has to produce a seeding graph for each machine. If we start with x_1, then the LS sub-algorithm yields the following line graph and machine seed set:

$$\begin{tikzcd} x_1 \arrow[r] & x_2 \arrow[r] & x_3 \arrow[r] & x_5 \arrow[r] & x_4 \end{tikzcd}$$

$$M_{S_{x_1}} = \{x_1, x_5\}.$$

Similarly, starting with x_2 yields:

$$\begin{tikzcd} x_2 \arrow[r] & x_1 \arrow[r] & x_3 \arrow[r] & x_5 \arrow[r] & x_4 \end{tikzcd}$$

$$M_{S_{x_2}} = \{x_2, x_5, x_1\}.$$

Starting with x_3 yields:

$$\begin{tikzcd} x_3 \arrow[r] & x_2 \arrow[r] & x_5 \arrow[r] & x_4 \arrow[r] & x_1 \end{tikzcd}$$

$$M_{S_{x_3}} = \{x_3, x_5, x_2\}.$$

Starting with x_4 yields:

$$\begin{tikzcd} x_4 \arrow[r] & x_2 \arrow[r] & x_5 \arrow[r] & x_1 \arrow[r] & x_3 \end{tikzcd}$$

$$M_{S_{x_4}} = \{x_4, x_2\}.$$

Finally, starting with x_5 yields:

$$\begin{tikzcd} x_5 \arrow[r] & x_1 \arrow[r] & x_2 \arrow[r] & x_3 \arrow[r] & x_4 \end{tikzcd}$$

$$M_{S_{x_5}} = \{x_5\}.$$

We let $\sum_{y \in M_{S_{x_i}}} J(y) := |M_{S_{x_i}}|$. The definition of K is irrelevant in this example because the sets $R_{S_{x_i}} = R \backslash M$ are equal for all $1 \leq i \leq 5$. Hence, for the cycle in Fig. 6, we select the seed where

$$\begin{aligned} M_S &= \{x_5\}, \text{ and} \\ R_S &= \{r_1, r_2 \backslash x_1, r_3\}. \end{aligned}$$

From an environmental standpoint, it is interesting to note how vital chlorine is to the cycle so that it always shows up in the machine seed set in one form or another.

V. Conclusions and Future Work

A novel algorithm to identify an optimal seed for a class of generation systems has been proposed. It utilizes the concepts of families and weak regularity to consider resources and their composition, deal with machines of deficient rank that are used as resources, isolate seed machines from generation cycles, and overcome the difficulty of seeding self-reproducing systems where the generation of a copy of a machine depends on the assistance of its offspring.

The avenues for future research include examining how one can control a generation system to produce an optimal seed. Once issues of control have been resolved, the ideal of finding a seed that can initiate an evolving self-reproducing system needs to be pursued. With the theory in place to analyze generation systems, the next step is to develop theory to synthesize generation systems.

The SI algorithm needs to be extended to 1) allow for the determination, whenever possible, of a seed of pre-specified order $\nu\mu$; 2) incorporate some notion of the quantity of a seed resource needed to perpetuate a system; and 3) recognize and compensate for time constraints that may impose a larger-size seed upon the system.

Appendix

Lemma 1.

Proof. We start with the definition of a strongly regular generation system: whenever $y = G(x, (r_{\mu_1}))$, where x and y are machines and (r_{μ_1}) is a sequence of μ_1 resources, we have $y \not\prec r$ for all $r \in (r_{\mu_1})$. Let x' denote the machine that produces machine y using resource r_{μ_1}, i.e., $y = G(x', r_{\mu_1})$. Since $y \not\prec r$ for all $r \in (r_{\mu_1})$, $y \not\prec r_{\mu_1}$. Thus, the definition of a weakly regular generation system is satisfied. □

Theorem 1.

Proof. The proof is by construction. Specifically, we outline an iterative algorithm that is guaranteed to identify a matriarch for a family. At the end of every iteration, the algorithm produces a partition of the family into a candidate matriarch, a set of descendants of that candidate matriarch, and a set of machines yet to be considered. During each iteration, the size of the set of machines yet to be considered is decreased by at least one unit, the size of the set of descendants of the candidate matriarch is increased by at least one unit, and the candidate matriarch itself may be updated. The algorithm terminates when the set of machines to be considered is empty, at which time the candidate matriarch is confirmed as a matriarch.

To initialize the algorithm, consider two arbitrary machines x and y of the family. Since x and y are in the family, they have a common ancestor z. We consider three cases:

(1) If $z = x$, then the candidate matriarch is x, the set of descendants of the candidate matriarch is the set of all machines obtained in the process of generating y from x (including y), and the initialization is complete.

(2) If $z = y$, then the candidate matriarch is y, the set of descendants of the candidate matriarch is the set of all machines obtained in the process of generating x from y (including x), and the initialization is complete.

(3) If z is neither x nor y, then the candidate matriarch is z, the set of descendants of the candidate is the set of all machines obtained in the process of generating both x and y from z (including x and y), and the initialization is complete.

Once the algorithm is initialized, each iteration proceeds as follows. Let x be the candidate matriarch, and consider an arbitrary machine y in the set of machines yet to be considered. Since x and y are in the family, they have a common ancestor z. We consider four cases:

(1) If $z = x$, then the candidate matriarch remains x, and all the machines obtained in the process of generating y from x (including y) are transferred into the set of descendants of the candidate matriarch and removed from the set of machines yet to be considered. This completes the iteration.

(2) If $z = y$, then the candidate matriarch becomes y, and all the machines obtained in the process of generating x from y (including x) are transferred into the set of descendants of the candidate matriarch and removed from the set of machines yet to be considered. This completes the iteration.

(3) If z is neither x nor y but is in the set of descendants of the candidate matriarch, then the candidate matriarch remains x, and all the machines obtained in the process of generating y from z (including y) are transferred into the set of descendants of the candidate matriarch and removed from the set of machines yet to be considered. This completes the iteration.

(4) If z is neither x nor y but is in the set of machines yet to be considered, then the candidate matriarch becomes z, and all the machines obtained in the process of generating both x and y from z (including x and y) are transferred into the set of descendants of the candidate matriarch and removed from the set of machines yet to be considered. This completes the iteration.

□

Theorem 2.

Proof. Weak connectivity of the directed graph representation of $\Gamma = (U, M, R, G)$ follows directly from the definition of a family. Indeed, since Γ is a family, for a particular $(x, y) \in M, \exists z \in M$, and $(r_n), (r_m) \in R$ such that $x = G(z, (r_n))$ and $y = G(z, (r_m))$. In the directed graph representation of Γ, there is a path from z to x through the sequence of edges labeled (r_n), and a path from z to y through the sequence of edges labeled (r_m). Hence, in the undirected version of this directed graph, there is a path from x to y via z. By the definition of weak connectivity, this means that x and y are weakly connected in the directed graph. Since this is true for all vertex pairs in the directed graph representation of a family, the entire graph is weakly connected. □

Theorem 3.

Proof. This proof follows directly from the definition of a seed. First, we are given that $y \prec (r_m)$. Since $\forall 1 \leq i \leq m$, $x_{i+1} = G(x_1, (r_i))$, a seed for the weakly regular family $(U, (x_{m+1}), (r_m), G)$ is:

$$\begin{aligned} S &= \{x_1\} \cup \{(r_m)\}; \\ &= \{x_1, y\} \cup \{(r_m) \backslash y\}. \end{aligned}$$

If $R_S \cap M_S = \oslash$, then we need to have $|M_S| \geq 1$ so that at least one machine is present to generate the system. We are given that x_1 can produce every machine in (x_{m+1}) using (r_m). From the seed set S above, since (r_m) contains only y but y cannot be generated by x_1, the system needs to be started with both x_1 and y. Therefore, $|M_S| = 2$, the minimum possible. □

Theorem 4.

Proof. This proof uses mathematical induction. We assume that the generation subsystem of our matriarch x_1, where $\Gamma_{x_1} = (U, M_{x_1}, R_{x_1}, G)$, is a DMST, and that we have selected a simple path in this tree that starts at x_1. Let M be the set of machines that are the vertices in this path, and R be the set of resources that are the edges in this path. Let $R_S = R \backslash M$, and $M_S = \{x_1\}$. Let r be the first resource edge in this path, and s be the second resource edge in this path. Let $y := G(G(x_1, r), s)$, which is different from $G(x_1, r)$ by the definition of a path.

Consider $G(x_1, r)$. If $r \succ y$, the sub-algorithm takes $M_S = M_S \cup \{y\}$, and by Theorem 3, the new M_S forms a seed for the path when united with R_S. Otherwise, the original M_S is still a seed when united with R_S, since y is not required.

For the induction hypothesis, assume that M_S forms a seed with R_S when x_1 uses a sequence of (r_{k-1}) resources. Let $y := G(G(x_1, (r_{k-1})), r_k)$, which is different from $G(x_1, (r_{k-1}))$ by the definition of a path.

Consider $G(x_1, (r_{k-1}))$. If $(r_{k-1}) \succ y$, the sub-algorithm takes $M_S = M_S \cup \{y\}$, and by Theorem 3, the new M_S forms a seed for the path when united with R_S. Otherwise, the original M_S is still a seed when united with R_S, since y is not required. □

Theorem 5.

Proof. We have to first prove that the output set S is a seed for the initial self-reproducing system. Since Γ is a union of families, and $S = \bigcup S_x$ for x belonging to the set of matriarchs $M_{\female}$, it suffices to prove that each S_x is a seed for one of the constituent families. Thus, we will show that each of the intermediate steps is correct.

By assumption, the generation system to be seeded is made up of one or more weakly regular families. The directed graph representation of a single family is weakly connected. Thus, the directed graph representation of the initial generation system is made up of one or more weakly connected components.

Each vertex in the directed graph representation belongs to a weakly connected component. Both the BFS or DFS algorithms are able to correctly find the vertices reachable from a root in a weakly connected directed graph.[16] Thus, the use of either of these algorithms ensures that this step is correct.

The SI algorithm considers a finite number of sets each with finite cardinality. There are several known algorithms that are able to correctly count the elements in a set and sort the sets in descending order. The use of any of these algorithms results in the selection process being correct.

To find the directed minimum spanning tree for the selected weakly connected component requires use of the Chu-Liu-Edmonds algorithm, or Tarjan's efficient implementation of the same. These algorithms have been proved to be correct.[20, 23, 24] We have shown by Theorem 4 that the LS sub-algorithm is correct for any path in the tree. Since the union of seed sets is itself a seed set, $S_x = \bigcup S_{path}$ is a valid seed. Just as in the previous step, there are known algorithms for correctly evaluating the sum of a functional on the elements of a set, sorting these sums, and picking the set with the minimum sum. The use of any of these algorithms results in the selection process being correct.

Therefore, S_x is a seed for all Γ_x.

We now demonstrate optimality.

Let $\Gamma = (U, M, R, G)$ be made up of k weakly regular disjoint families. By assumption, all the machines in the given generation system need to be produced. Hence, the optimal seed for each family must include the least costly resources such that all machines in the family are generated. This implies that there must exist a path between the root vertex and all other vertices in the directed graph representation of the subsystem of a matriarch, and the DMST that is found via the Chu-Liu-Edmonds algorithm satisfies this property with minimal cost. We have shown that each pass through the SI algorithm produces a seed for a family, before the family is removed from the original generation system. By Theorem 3, this seed contains the minimum number of machines to generate each path. For paths with a common sub-path, the minimum number of machines to generate the common sub-path is the same and is unaffected by the union operation. For disjoint paths, the minimum number of machines to generate the paths is the sum of the minimum number of machines to generate each path, which is the number of machines produced by the union operation. Thus, the number of machines in the machine seed set of a matriarch is a minimum for each family if that matriarch begins the generation process. The seed set chosen for the family is the set with lowest total cost of machines and resources, and is selected from all the possible matriarch seed sets for that family. If there are k disjoint families in the original system, the SI algorithm will iterate k times before returning a seed that is the union of the seed sets for each family. Taking all such minimal cost seeds produces an optimal seed set for each family, and since the families are disjoint, the union of these sets results in a seed for the original generation system that is optimal with respect to cost. $\square$

Theorem 6.

Proof. We have to show that if a seed exists, the algorithm in this paper will output one possible seed. Consider that a seed for a generation system always exists − this is the trivial seed, consisting of all the machines and resources in the generation system, i.e., $S = M \cup R$. Indeed, the algorithm presumes this seed at the start, before removing redundant resources and machines that belong to a matriarch's subsystem. Theorem 5 shows that the output of the algorithm is a seed.

Thus, completeness is guaranteed. $\square$

Theorem 7.

Proof. The LS sub-algorithm is convergent because no circuits exist in the DMST, and there are a finite number of paths of finite length that begin at the root of the tree. Each iteration of the SI algorithm removes elements from a set with finite cardinality, and this algorithm stops once the set is depleted.

We now consider the time complexity of operation during the first iteration of the algorithm. Let $|M| = n$ and $|R| = m$.

The use of either one of the BFS or DFS algorithms has time complexity $O(n + m)$.[16]

The fact that each machine has to be visited in order to determine the cardinality of the machine set of its generation subsystem results in a time complexity of $O(n)$.

The time complexity of the DMST algorithm is $O(n_p m_p)$,[19] where n_p is the number of machines in the primary subsystem, and m_p is the number of resources in the primary subsystem. The use of the DFS algorithm to identify the simple paths in the DMST has time complexity $O(n + m)$. The LS sub-algorithm visits all the machines in a simple path once, and this is repeated for a finite number of simple paths. The fact that (in the worst case) all primary subsystem resources have to be visited in order to remove any contained machines results in a time complexity of $O(m_p)$. Accounting for the possibility that there is more than one matriarch to apply the DMST algorithm to, this entire step could be repeated $n_♀$ times, where $n_♀$ is the number of matriarchs. Thus this step has polynomial time complexity.

All primary subsystem machines have to be removed from the original machine set, so that the time complexity of this step is $O(n_p)$.

Thus, the overall time complexity during the first iteration of the algorithm is of polynomial order in n and m. $\square$

References

[1] von Neumann, J., *Theory of Self-Reproducing Automata*, University of Illinois Press, 1966.

[2] Freitas Jr., R. A. and Merkle, R. C., *Kinematic Self-Replicating Machines*, Landes Bioscience, 2004.

[3] Sipper, M., "Fifty Years of Research on Self-Replication: An Overview," *Artifical Life*, Vol. 4, No. 3, 1998, pp. 237–257.

[4] Owens, P. and Ulsoy, A. G., "Self-Replicating Machines: Preventing Degeneracy," Tech. Rep. CGR-06-02, The University of Michigan, 2006.

[5] Kabamba, P., "The von Neumann Threshold of Self-Reproducing Systems: Theory and Computation," Tech. Rep. CGR-06-11, The University of Michigan, 2006.

[6] Menezes, A. and Kabamba, P., "Information Requirements for Self-Reproducing Systems in Lunar Robotic Colonies," *Proceedings of the 57th International Astronautical Congress*, No. IAC-06-A5.P.04, 2-6 October 2006.

[7] Menezes, A. and Kabamba, P., "A Combined Seed-Identification and Generation Analysis Algorithm for Self-Reproducing Systems," *Proceedings of the 2007 American Control Conference*, 11-13 July 2007, pp. 2582–2587.

[8] Menezes, A. and Kabamba, P., "An Optimal-Seed Identification Algorithm for Self-Reproducing Systems," *Proceedings of the 58th International Astronautical Congress*, No. IAC-07-D3.2.02, 24-28 September 2007.

[9] Foing, B., "Roadmap for Robotic and Human Exploration of the Moon and Beyond," *Proceedings of the 56th International Astronautical Congress*, No. IAC-05-A5.1.01, 17-21 October 2005.

[10] Wertz, J. R. and Larson, W. J., editors, *Space Mission Analysis and Design*, Microcosm Press, 3rd ed., 1999.

[11] Lodish, H., Berk, A., Matsudaira, P., Kaiser, C. A., Krieger, M., Scott, M. P., Zipursky, L., and Darnell, J., *Molecular Cell Biology*, W. H. Freeman, 5th ed., 2004.

[12] Hobbs, P. V., *Introduction to Atmospheric Chemistry*, Cambridge University Press, 2000.

[13] Adey, W. H. and Loveland, K., *Dynamic Aquaria*, Academic Press, 2nd ed., 1998.

[14] Menezes, A. and Kabamba, P., "On the Seeding of Self-Reproducing Systems," Tech. Rep. CGR-07-08, The University of Michigan, 2007.

[15] Diestel, R., *Graph Theory*, Springer-Verlag Heidelberg, 3rd ed., 2005.

[16] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C., *Introduction to Algorithms*, The MIT Press, 2nd ed., 2001.

[17] Aho, A. V., Hopcraft, J. E., and Ullman, J. D., *The Design and Analysis of Computer Algorithms*, Addison-Wesley Publishing Company, 1974.

[18] Hopcroft, J. and Tarjan, R., "Algorithm 447: Efficient Algorithms for Graph Manipulation," *Communications of the ACM*, Vol. 16, No. 6, 1973, pp. 372–378.

[19] Even, S., *Graph Algorithms*, Computer Science Press, 1979.

[20] Tarjan, R. E., "Finding Optimum Branchings," *Networks*, Vol. 7, 1977, pp. 25–35.

[21] Chirikjian, G. S., Zhou, Y., and Suthakorn, J., "Self-Replicating Robots for Lunar Development," *IEEE/ASME Transactions on Mechatronics*, Vol. 7, No. 4, Dec. 2002.

[22] Suthakorn, J., Kwon, Y. T., and Chirikjian, G. S., "A Semi-Autonomous Replicating Robotic System," *Proceedings of the 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, July 2003.

[23] Chu, Y. J. and Liu, T. H., "On the Shortest Arborescence of a Directed Graph," *Scientia Sinica*, Vol. 14, 1965, pp. 1396–1400.

[24] Edmonds, J., "Optimum Branchings," *Journal of Research of the National Bureau of Standards*, Vol. 71B, No. 4, 1967, pp. 233–240.