Amor Menezes* and Pierre Kabamba†

AN OPTIMAL-SEED IDENTIFICATION ALGORITHM FOR SELF-REPRODUCING SYSTEMS

Department of Aerospace Engineering
University of Michigan
Ann Arbor, Michigan, 48109-2140
United States of America
amenezes@umich.edu, kabamba@umich.edu

Abstract

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 endowed with the capacity for self-reproduction. An important consideration for the establishment of a self-reproducing robotic colony is the identification of a seed. This paper outlines a novel algorithm to determine the optimal seed for a class of self-reproducing systems, and illustrates the application of the algorithm on a modified version of a self-replicating system documented in the literature.

The technical approach of this paper utilizes concepts from Generation Theory. Self-reproduction is achieved by the actions of a robot on available resources, and so a seed for the colony consists of a set of robots and a set of resources. In a previous approach to the seeding problem, a Seed Identification and Generation Analysis algorithm included certain restrictive assumptions about the self-reproducing system under consideration, finding possibly non-optimal seeds.

The original contribution of this paper is to provide an algorithm that finds an *optimal* seed for a more general class of self-reproducing systems. The relationship between the size of the seed and the characteristics of a self-reproducing system is also investigated. Further, the necessary and sufficient conditions to produce an optimal seed are developed. An illustration of the algorithm's operation is provided.

This work is applicable to any system that requires adaptable robots be operated in a resource-constrained environment. It is expected that this paper will assist in moving from analyzing self-reproducing systems to synthesizing them.

1 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 [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 in [4] indicates that no one had published an evaluation of this threshold in the following 60 years. Recently, [5] developed a novel theory of generation that is able to compute this von Neumann threshold. The results in [5] included a necessary and sufficient condition for non-degenerate offspring, i.e., offspring with the same reproductive capability as the progenitor. Reference [6] presented a probabilistic version of these results, and also demonstrated parallels with information theory. The paper in [7] extended these results by providing an algorithm that identified a seed for a particular class of self-reproducing systems. This work builds on the previous paper by providing a seeding algorithm that is applicable to a more general class of self-reproducing systems. Additionally, it is demonstrated that the output of

*Ph.D. Candidate, Dept. of Aerospace Engineering.
†Professor, Dept. of Aerospace Engineering.

this algorithm is optimal.

The remainder of this section presents a rationale for the identification of a seed for a self-reproducing system, surveys background material on Generation Theory [5], discusses what makes the general seeding problem difficult, and highlights the results of [7] that serve as a foundation for the rest of this paper. Section 2 details the necessary assumptions, definitions, and methodology for seed identification, and outlines a Seed Identification (SI) algorithm. The properties of the SI algorithm are also analyzed in this section. Section 3 illustrates the application of the algorithm to a modified version of a self-replicating system documented in [8] and [9].

1.1 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 [10] is typical, suggest that individual countries will deploy advanced robots on an as-needed basis 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 [11]. 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 technology 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.

In order to minimize mass, it would be even more efficient to recognize the required elements for the initiation of a self-reproducing system, and send the smallest quantity of these elements into space. The identification of this minimal "seed" is the goal of this paper.

1.2 Highlights of Generation Theory

We first state what is meant by the following terms that will be used throughout the paper: reproduction, replication, self-reproduction, and self-replication. For a historical perspective of the first two terms, the reader is referred to Freitas' excellent discussion on the subject in [2]. We consider reproduction in biological systems to imply the capacity for genetic mutations and the potential for evolution. Thus from an information standpoint, reproduction involves a change to the DNA code during the generation of progeny. Likewise, we will take *reproduction* in an artificial system to imply a change in the information specifications of an offspring. We reserve the term *replication* for progeny that have identical information content to that of the progenitor. *Self-reproducing* and *self-replicating* will be used to refer to those entities that perform the information equivalent of asexual reproduction or mitosis, i.e., the entities can reproduce or replicate based on the information specifications of only one progenitor.

The theory that is surveyed here formalizes self-reproduction by "machines," a term describing any entity that is capable of producing an offspring regardless of its physical nature. Thus 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 *in the context described;*
- $R \subseteq U$ *is a* set of resources *that can be utilized for self-reproduction; and,*
- $G : M \times R \rightarrow U$ *is a* generation function *that maps a machine and a resource into an out-*

come in the universal set, and not necessarily in the set of machines.

Furthermore, 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 belong to the set of resources, and the latter states that outcomes of attempts at generation may be neither machines nor resources.

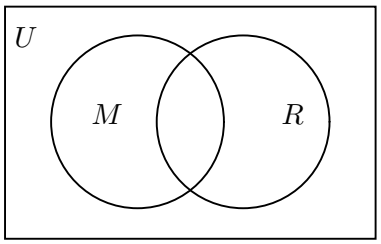

Figure 1: Pictorial representation of Definition 1.

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

We make use of concepts from graph theory [12] in this paper. Equation (1) may be represented by a *directed reproduction graph*, γ, as shown in Figure 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.

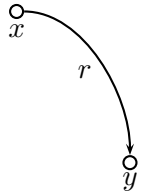

Figure 2: The directed reproduction graph of (1).

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 equal to the binary relation (M, M, γ).*

Definition 3. *The* generation sets *in a generation system are defined as:*

- *$M_0 = M$, the set of all machines;*
- *M_{i+1}, the set of all machines that are capable of producing a machine of M_i, $\forall i \geq 0$. That is, for $x \in M_{i+1}$, $\exists\ y \in M_i$ such that $y = G(x, r)$.*

These sets are nested with the innermost generation set being important for self-reproduction. This set can be defined as:

$$M_\infty = \bigcap_{i=0}^{\infty} M_i. \tag{2}$$

It is shown in [5] that generation always proceeds outwards. Also, the notion of the rank of a generation system, as defined below, is emphasized.

Definition 4. *The* rank of a generation system, *$\rho(\Gamma)$, where $\Gamma = (U, M, R, G)$ with generation sets M_i, $i \geq 0$, is the smallest integer ρ such that $M_\rho = M_{\rho+1}$. If $\forall i, M_i \neq M_{i+1}$, then the generation system has infinite rank.*

For a generation system of finite rank ρ, the nesting of the generation sets stop at the integer ρ. All generation sets of order greater than ρ (up to and including M_∞) are equal. A generation system that has a finite number of machines always has finite rank.

Definition 5. *The* rank of a machine, *$\rho(x)$, in a generation system $\Gamma = (U, M, R, G)$ with generation sets M_i, $i \geq 0$, and $\rho(\Gamma) = \rho$, is equal to i if $x \in M_i \backslash M_{i+1}$ ("deficient generation rank"), or is equal to ρ if $x \in \bigcap_{i=0}^{\infty} M_i$ ("full generation rank").*

Definition 6. *A* generation cycle *is a sequence of generations resulting in the production of a machine identical to itself after n generations.*

Machines capable of replication (a generation cycle of order one) in a generation system must belong to M_∞, and any exit from M_∞ is irreversible. It is possible for offspring machines to belong to M_∞

as long as their progenitors do as well. Thus the requirements for non-degenerate reproduction and replication are quantified. It is proved in [5] that there is a minimum threshold of rank above which a machine is able to generate an offspring without a decrease in generation rank. We call this the *von Neumann Rank Threshold*, τ_r, and define

$$\tau_r = \rho(\Gamma). \tag{3}$$

The reader is referred to the material in [5] for proofs of the above statements, as well as many other insights into the information requirements of self-reproducing systems.

1.3 The Seeding Problem

There are many factors that contribute to the inherent difficulty of seeding, including:

(a) the possibility that a given generation system is made up of multiple, distinct subsystems, each with a different seed. Alternatively, there could be multiple, indistinct subsystems, with some common seed elements for 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 within a given self-reproducing system. If these cycles exist, then one naturally wonders which of the machines in a particular loop should be chosen to belong to the seed.

(c) the fact that degenerate machines, i.e., those that have deficient machine rank, should not belong to the seed for a self-reproducing system. This is because there will be some future time when their progeny can no longer produce offspring that are machines. On the other hand, if the whole system is degenerate, and there does not exist a machine with full generation rank, then there is a need to identify the machine that is the least degenerate.

(d) the complexity of the resource set. A consistent theme in the literature of self-reproducing systems is that a machine does not operate on one resource element during a generation attempt, but rather on an ordered list of elements that constitute a particular 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 other machines, degenerate or otherwise, since other machines can be cannibalized or act as catalysts for generation. Thus, each resource needs to be carefully scrutinized in order to determine the contents of 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 attempt at tackling a restricted version of this problem for a small class of generation systems, [7], resulting in the Seed Identification and Generation Analysis (SIGA) algorithm. We now summarize some of the assumptions and simplifications that we had made to help structure the problem, but which, unfortunately, made non-optimal seeds possible. The remainder of the paper builds on this summary.

1.4 A Prior Solution to Seeding

In [7], assumptions were made with respect to both machines and resources. We first assumed that every resource in the set R of a generation system was utilized by a progenitor machine so that another machine could be produced.

Assumption 1. *Given a generation system* $\Gamma = (U, M, R, G)$*, we assume that* $\forall r \in R, \exists x \in M$ *such that* $G(x, r) \in M$*.*

This assumption simplified the selection procedure of resources since it pointed to the condition that all resources were necessary to produce an offspring. Hence, a seeding algorithm could simply identify all possible resources as constituents of a seed. If we accepted that all resources were necessary however, then we allowed ourselves the possibility of selecting redundant resources. For instance, if there existed two resources such that a progenitor machine would produce the same offspring with each of those two resources, then by taking both resources to belong to the seed, a redundant selection was made and the resulting seed was non-optimal. We ignored this possibility and considered it an avenue for future refinement.

We allowed each resource to itself contain an ordered list of physical elements that could include

machines. We therefore defined a containment relation as follows.

Definition 7. *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.

Of course, if machine x_i was a resource itself, then this relation still held true. We needed the following definition as well, before making an assumption on resources that has been validated by all literature to date.

Definition 8. *If machines $x_1, x_2, \ldots, x_\nu$ are contained in resource r, then we use the notation $r\backslash(x_1, x_2, \ldots, x_\nu)$ to refer to an ordered list of the elements of r that does not contain the machines $x_1, x_2, \ldots, x_\nu$.*

Assumption 2. *Given a generation system $\Gamma = (U, M, R, G)$, we assume that 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., $r\backslash x \in R$.*

Next, we assumed that every machine in the generation system had a progenitor machine.

Definition 9. *A* surjective generation system *is a generation system $\Gamma = (U, M, R, G)$ where $\forall y \in M, \exists x \in M$, and $\exists r \in R$ such that $y = G(x, r)$.*

We further assumed that there existed a machine in the generation system that was capable of producing any machine in the system after μ generations. This was a special case of a surjective generation system.

Assumption 3. *We assume that in the generation system $\Gamma = (U, M, R, G)$, $\exists x_0 \in M$ such that $\forall x_1 \in M, \exists \mu_1 \leq \mu, \exists r_1, r_2, \ldots, r_{\mu_1}$ selected from R such that*

$$G(\ldots G(G(G(x_0, r_1), r_2), r_3) \ldots, r_{\mu_1}) = x_1.$$

It was this rather restrictive assumption that, in part, gave us the SIGA algorithm. The approach to developing that seed identification algorithm was similar to the Generation Analysis Algorithm (GAA) stated in [5], and in fact utilized the GAA in its operation. The GAA employs the concept of an *outer layer*, first introduced in [5] and defined as follows.

Definition 10. *The* outer layer *of a generation system $\Gamma = (U, M, R, G)$ is the set $M_0\backslash M_1$. This is the set of machines such that, no matter what resource they use, they produce an offspring that is no longer a machine, i.e.,*

$$\{x \in M : \forall r \in R, G(x, r) \notin M\}.$$

After an outer layer is removed, a generation system of reduced rank remains. The GAA works by peeling away the outer layers of each of the generation systems Γ_i, $0 \leq i \leq \rho$. We applied a similar notion to the development of the SIGA algorithm, having that algorithm peel away outer layers in both the set of machines and the set of resources, before picking one machine and all reduced resources to belong to the seed set (see [7] for the full algorithm methodology, pseudocode, and an example application).

2 Seed Identification

For a more general approach to the seeding problem, we consider an arbitrary generation system. The number of distinct machines is finite in this system, and the number of distinct resources is finite as well. Just as von Neumann postulated automata that self-reproduce in a "sea of parts" [1], we assume that an inexhaustible supply of each resource exists. Another common assumption is that all the machines in the generation system need to be produced, even though they need not all belong to the seed set. Of the assumptions listed previously, we require that Assumption 2 hold for this system also. Typically, the mass of the resources must be optimized. Since the machines in a generation system are often of similar mass, it is frequently specified that the quantity of machines, rather than the mass, is also minimized.

The given self-reproducing system may not be surjective, and even if it is, we may not be able to use the SIGA algorithm because Assumption 3 may fail. To formulate seeding requirements in a mathematically precise way, we begin by defining a seed, selecting the class of generation systems that we will deal with, and then using the properties of these systems to help setup the seeding problem.

2.1 Problem Definition and Setup

We will use a more compact notation to denote the sequential selection of resources. Let (r_{μ_1}) be a sequence of μ_1 resources from R, so that

$$G\left(x_0, (r_{\mu_1})\right) := G(\ldots G(G(x_0, r_1), r_2)\ldots, r_{\mu_1}).$$

Then the general definition of a seed is as follows.

Definition 11. *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

$$G(y_0, (r_{\mu_1})) = y_1.$$

In the rest of the paper, we design an algorithm to produce a seed as per the above definition. The idea is to reduce certain generation systems into a form that can be easily seeded. We need a few more definitions before we can indicate the type of generation systems our algorithm is restricted to.

Definition 12. *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, no machine can be contained in its ancestry.

Definition 13. *A* family *is a generation system $\Gamma = (U, M, R, G)$ where $\forall (x, y) \in M, \exists z \in M$, and $(r_n), (r_m) \in R$ such that $x = G(z, (r_n))$ and $y = G(z, (r_m))$. A* matriarch *of a family is an element $x_0 \in M$ such that $\forall x_1 \in M, x_1 \neq x_0, \exists (r_{\mu_1})$ selected from R such that $G(x_1, (r_{\mu_1})) = x_0$.*

Note that empty sequences are allowed in the above definition of a family, so that it is possible to consider either x or y to be the common ancestor z.

Proposition 1. *Every family $\Gamma = (U, M, R, G)$ has a matriarch.*

Proof. See Appendix. □

Proposition 2. *The directed graph representation of a family $\Gamma = (U, M, R, G)$ is weakly connected.*

Proof. See Appendix. □

The converse to Proposition 2 is not true: a weakly connected digraph does not imply that the represented generation system is a family (see Figure 3 for an example).

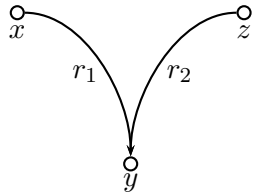

Figure 3: A weakly connected digraph representing two families.

Assumption 4. *We assume that the generation system to be seeded, $\Gamma = (U, M, R, G)$, is strongly regular and made up of one or more disjoint families.*

The nature of the generation system in Assumption 4 helps setup the seeding problem when we note that the underlying undirected graph of a family is connected. Since the connected components of a graph are the equivalence classes of the path existence relation between two vertices [12], and the directed graph representation of the generation system to be seeded is made up of one or more connected components, we can partition this graph into its connected components. Thus, for the class of systems in Assumption 4, 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 14. *The* generation subsystem of machine x_0 *is the generation system*

$$\Gamma_{x_0} = (U, M_{x_0}, R_{x_0}, G)$$

where,

$$\begin{aligned} M_{x_0} &= \bigcup_{i=0}^{\infty} M_{x_0}^i \\ M_{x_0}^i &= \{x \in M : \exists (r_i) \in R : x = G(x_0, (r_i))\} \\ R_{x_0} &= \bigcup_{i=0}^{\infty} R_{x_0}^i \\ R_{x_0}^i &= \bigcup \{(r_i) \in R : G(x_0, (r_i)) \in M\}. \end{aligned}$$

In Definition 14, $M_{x_0}^i$ is the set of all the descendants of x_0 produced after i generations, M_{x_0} is the set of all the descendants of x_0, $R_{x_0}^i$ is the set of all resource sequences of length i that would produce a descendant of x_0, and R_{x_0} is the set of all resource sequences that would produce a descendant of x_0. Hence, the generation subsystem of x_0 is the largest family for which x_0 is a matriarch.

The idea for the SI algorithm is to 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. We will let M_{φ} denote the set of matriarchs.

In the generation system of a matriarch, x_0, every machine in the subsystem can be produced except possibly x_0 itself. Thus, in the course of seeding the subsystem of x_0, the machine to pick for the seed set of the subsystem, S_{x_0}, is x_0. The rationale for this process of identifying one machine of highest rank that can generate every machine in its subsystem comes from the next two propositions.

A necessary condition to minimize $|M_S|$ is the following.

Proposition 3. *Let $\Gamma = (U, M, R, G)$ be a family, and S be a seed set of Γ for which $|M_S|$ is a minimum. Then $\forall x \in M_S$,*

$$\rho(x) = \begin{cases} \rho, & \text{if } |M_\infty| > 0; \\ \max_{y \in M} \rho(y), & \text{if } |M_\infty| = 0. \end{cases}$$

Proof. See Appendix. □

Corollary 1. *If $\Gamma = (U, M, R, G)$ is a family, and $|M_\infty| > 0$, then $|M_S| \leq |M_\infty|$. This is because an optimal M_S can only include machines from M_∞. On the other hand, if $|M_\infty| = 0$, then an optimal M_S has $|M_S| = 1$. This is because there is only one machine for which the maximum rank condition is satisfied, a consequence of the fact that generation always proceeds outwards [5].*

A sufficient condition to minimize $|M_S|$ is the following.

Proposition 4. *Assume that the generation subsystem of machine x, $\Gamma_x = (U, M_x, R_x, G)$ is strongly regular. Then a seed set for Γ_x, S_x, where $M_{S_x} = \{x\}$ and $R_{S_x} = R_x \backslash M_x$ has the minimum $|M_{S_x}|$.*

Proof. See Appendix. □

In a strongly regular family, if a machine is contained in a resource, then that resource cannot be utilized in any sequence of resources used to generate the machine. Hence, the notion of containment has no effect on the seeding process for these systems. This is why we restrict the class of generation systems in this paper to those that are solely made up of strongly regular families.

With these preliminaries out of the way, we can concentrate on the actual seeding and partitioning process.

2.2 The SI Algorithm

Assuming the given self-reproducing system of n machines and m resources is strongly regular and made up of one or more disjoint families, the first step of the algorithm would be to find the generation subsystems of all the machines in M, i.e.,

Step 1

For all x_i in M, $1 \leq i \leq n$, determine Γ_{x_i}.

Since we can represent a generation system as a weighted, directed graph, we can use established notions in graph theory to aid us in the subsystem identification process. 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 [13, 14, 15, 16]. Applying either of these algorithms to the graph of the generation system in Step 1 yields Γ_{x_i} for all $1 \leq i \leq n$. In these subsystems, there is one machine capable of generating all machines in M_{x_i} except possibly x_i itself. We now want to partition the initial generation system in the following manner.

Step 2

Select the Γ_{x_i} where $|M_{x_i}| \geq |M_{x_j}|$, $\forall 1 \leq j \leq n$, by counting the cardinalities of the machine sets and sorting these sets.

This Γ_{x_i} is the largest generation subsystem of the initial self-reproducing system. We consider this to be our primary generation subsystem, and regard the system where the machine set is $M \backslash M_{x_i}$ to be a secondary generation subsystem. The secondary subsystem requires the removal of all $x \in M_{x_i}$ from M. The idea is to seed our primary subsystem first, and then go back to the secondary subsystem and partition and seed iteratively.

In Step 2, if there are two subsystems Γ_{x_i} and Γ_{x_j} with the same machine set, then both x_i and x_j are matriarchs for the same family. To ensure the optimal seeding of this family, we will need to compare the cost of the seed resources when the subsystem of x_i is a primary subsystem and when the subsystem of x_j is a primary subsystem.

However, before we can tackle seeding of a primary subsystem (and by extension, the seeding of all other partitions), we need to ensure that the subsystem under consideration has the property that each offspring is generated from only one resource. Thereafter, if we select all resources to be a part of the seed set for the subsystem, we have avoided any unnecessary selection of redundant resources.

Let $J : R \rightarrow \mathbb{R}$ be a cost functional representing the mass of a resource, or the quantity required of a resource, or the resource's availability, etc. We will again make use of a result in graph theory for the next step. In graph theory, a subgraph of a finite directed graph is called a *branching* if it has the following properties: it contains all the vertices of the original graph (spanning); it is circuit-free; and the number of edges entering any vertex is less than or equal to one. If the number of entering edges is zero for only one of the subgraph's vertices, r, and the remaining vertices all have only one edge entering them, then the branching is a directed tree with root r [15]. The problem of finding a branching for which the sum of the edge costs is optimal (a maximum) was solved independently in [17, 18, 19], is well-treated in [13, 15, 20], and can be efficiently implemented using [21].

We have a similar situation if we first add a new machine, x_i', so that all instances of the resources (edges) that are used to produce machine x_i (enter the root vertex) in the primary subsystem (directed graph) are now used to produce machine x_i'. We can then use the Chu-Liu-Edmonds algorithm to obtain a generation subsystem where Proposition 4 is still applicable, but where redundant resources are also not present.

The implementation of the optimal branching algorithm that is assumed requires that: 1) the directed spanning tree that is found has a minimum rather than a maximum cost, and 2) a root vertex is accepted as additional input, so that search for the tree starts from this root instead of the first entry in a vertex-edge incidence list. We utilize these notions in formulating Step 3 of the SI algorithm.

Step 3

for all the matriarchs of the largest generation subsystem **do**
  if in the graph representation of Γ_{x_i}, x_i has entering edges **then**
    Add a new vertex x_i'.
    Change these edges so that they now enter x_i'.
  end if
  Find the directed minimum spanning tree (DMST) in the graph of Γ_{x_i} with root at x_i.
  $\Gamma_{x_i min} \leftarrow$ the DMST of Γ_{x_i}.
end for
Select the $\Gamma_{x_i min}$ for which $\sum_{r \in R_{x_i min}} J(r)$ is a minimum.

We can now seed the resultant self-reproducing subsystem.

Step 4

$S_{x_i} = \{x_i\} \cup (R_{x_i min} \backslash M_{x_i})$.

Next, we obtain the generation system that remains to be seeded.

Step 5

Remove all $x \in M_{x_i}$ from M.

We continue the process so far on the subsidiary generation subsystems, iterating from Step 2 until there are no more machines left in M. The entire seed set is the union of all the seed sets for the various generation subsystems.

Step 6

if $M \neq \oslash$ **then**
  Go to Step 2.
else
  $S \leftarrow \bigcup S_{x_i}$.
  Stop.
end if

2.3 Properties of the SI Algorithm

In this subsection, we make some claims about the Seed Identification algorithm and the resultant seed that is output. The proofs of these claims can be found in the Appendix.

Proposition 5. *The SI algorithm is correct. That is, the output of the algorithm is a seed for the given generation system.*

Proof. See Appendix. □

Proposition 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. □

Proposition 7. *The SI algorithm is guaranteed to stop after a finite number of iterations. The best-case time complexity for the operation of this algorithm is $O(3n + nm + 2m + 1)$ iterations. The worst-case time complexity for the operation of this algorithm is the worst of $O(n^2m + 4n + 2m)$ and $O(n^2 + 5n + m)$ iterations.*

Proof. See Appendix. □

Proposition 8. *The SI algorithm produces a seed that is optimal with respect to the number of machines and the cost of the resources in the seed.*

Proof. See Appendix. □

Proposition 9. *Given a family $\Gamma = (U, M, R, G)$, the size of the seed either increases or stays constant with expanding M or R.*

Proof. See Appendix. □

3 An Example Application of the SI Algorithm

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. [8, 9].

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

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

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.

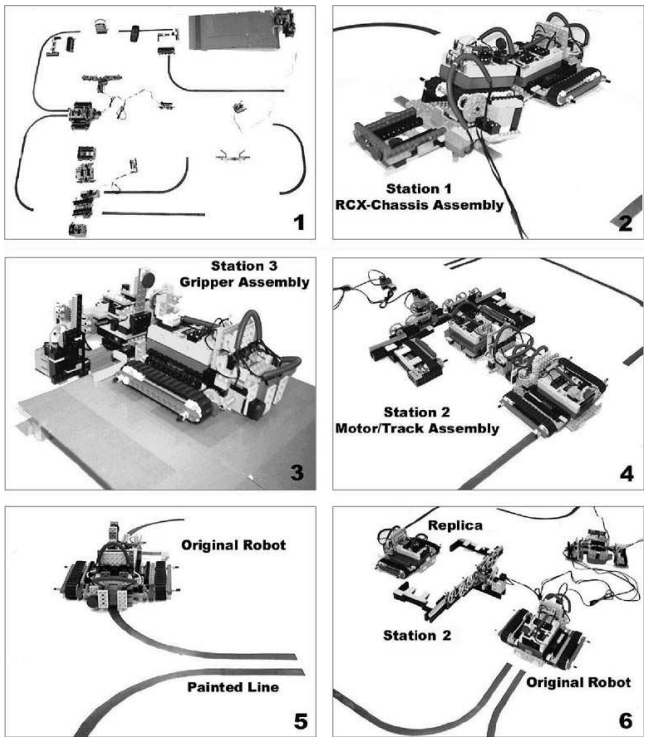

Figure 4: The semi-autonomous replication process of the Suthakorn-Kwon-Chirikjian robot [8].

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 above is a single family, and so the application of 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.

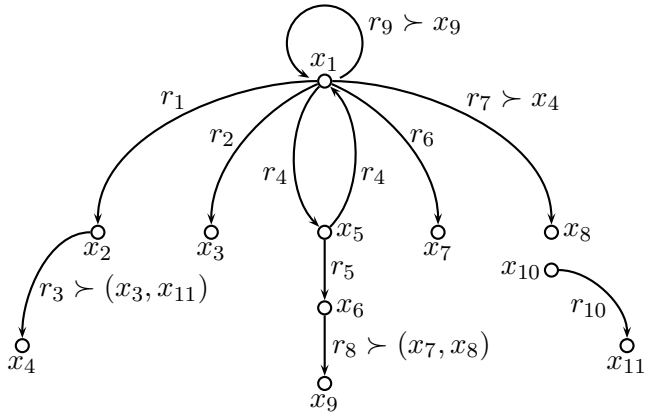

Figure 5: Directed graph representation of the modified Suthakorn-Kwon-Chirikjian semi-autonomous replicating system.

This generation system is strongly regular, and is made up of two disjoint families. 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 at each step.

Step 1: The machine sets of $\Gamma_{x_i}, 1 \leq i \leq 11$ are the following.

$M_{x_1} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$

$M_{x_2} = \{x_4\}$

$M_{x_3} = \oslash$

$M_{x_4} = \oslash$

$M_{x_5} = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$

$M_{x_6} = \{x_9\}$

$M_{x_7} = \oslash$

$M_{x_8} = \oslash$

$M_{x_9} = \oslash$

$M_{x_{10}} = \{x_{11}\}$

$M_{x_{11}} = \oslash$

Step 2: 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.

Step 3: For x_1,

$R_{x_1 min} = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8, r_9\}$.

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.

For x_5, $R_{x_5 min} = \{r_1, r_2, r_3, r_4, r_5, r_6, r_7, r_8\}$.

The DMST algorithm does not select r_9.

$\Gamma_{x_5 min}$ is selected.

Step 4: For the first family, we get

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

Step 5: We are left with $M = \{x_{10}, x_{11}\}$.

Step 6: We now go back to Step 2.

Step 2: We select x_{10}.

Step 3: $R_{x_{10} min} = \{r_{10}\}$.

Step 4: For the second family we get

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

Step 5: We are left with $M = \oslash$.

Step 6: A seed for this system is

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

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.

4 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 strong regularity to consider resources and their composition, deal with machines of deficient rank that are used as resources, and isolate seed machines from generation cycles. We have developed the necessary and sufficient conditions to produce an optimal seed, and investigated the relationship between the size of the seed and the characteristics of a generation system.

The avenues for future research include relaxing the requirement for strong regularity, and 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 or possible non-existence of a seed of order $\nu\mu$, with $\nu\mu$ pre-specified; 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. These three apparent limitations will be overcome in future work.

Appendix

Proposition 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 al-

gorithm 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.

□

Proposition 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. □

Proposition 3.

Proof. This proof is by contradiction. Let $|M_S|$ be a minimum.

Case 1: $|M_\infty| > 0$.

Suppose that $\exists x \in M_S$ such that $\rho(x) < \rho$. From Generation Theory [5], since Γ is a family, $\exists y \in M, r \in R$ such that $x = G(y, r)$, and $\rho(x) < \rho(y) \leq \rho$.

From the definition of a seed (Definition 11), $\exists z \in M_S, (r_n) \in R_S$ such that $y = G(z, (r_n))$.

Thus, both z and x belong to M_S.

Let $(r_m) := ((r_n), r)$, so that $(r_m) \in R$.

Then $G(z, (r_m)) = x$, and so $S' = (M_S \backslash \{x\}) \cup R_S$ is a valid seed.

But $|M_S \backslash \{x\}| < |M_S|$, and so $|M_S|$ is not a minimum, a contradiction.

Case 2: $|M_\infty| = 0$.

Suppose that $\exists x \in M_S$ such that $\rho(x)$ is not the maximum over all machines in the family. From Generation Theory [5], since Γ is a family, $\exists y \in M, r \in R$ such that $x = G(y, r)$, and $\rho(x) < \rho(y) < \rho$.

From the definition of a seed (Definition 11), $\exists z \in M_S, (r_n) \in R_S$ such that $y = G(z, (r_n))$.

Thus, both z and x belong to M_S.

Let $(r_m) := ((r_n), r)$, so that $(r_m) \in R$.

Then $G(z, (r_m)) = x$, and so $S' = (M_S \backslash \{x\}) \cup R_S$ is a valid seed.

But $|M_S \backslash \{x\}| < |M_S|$, and so $|M_S|$ is not a minimum, a contradiction.

□

Proposition 4.

Proof. This proof follows directly from the definitions of strong regularity and generation subsystem. Indeed, if $R_{S_x} \cap M_x = \oslash$, then we need to have $|M_{S_x}| \geq 1$ so that at least one machine is present to generate the system. Since Γ_x is the generation subsystem of x, x can generate every machine in M_x by definition. Since Γ_x is strongly regular, any resources that contain machines cannot be used to generate those machines, by definition. This implies that machines additional to x are not needed. Therefore, the set $S_x = \{x\} \cup (R_x \backslash M_x)$ is a valid seed.

Moreover, $|M_{S_x}| = 1$, the minimum possible. □

Proposition 5.

Proof. We have to 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 $M_{\mathbb{Q}}$, it suffices to prove that each S_x is a seed for one of the constituent families. Thus, we will show that each of Steps 1 through 4 is correct.

Step 1.

By assumption, the generation system to be seeded is made up of one or more strongly 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 [13]. Thus, the use of either of these algorithms ensures that this step is correct.

Step 2.

Here, 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.

Step 3.

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 [17, 18, 21]. Just as in Step 2, 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.

Step 4.

This part of the proof is similar to the proof of Proposition 4. Since Γ_{xmin} is the generation subsystem of x with the added property that each offspring is generated from only one resource, x can generate every machine in $M_{xmin} = M_x$ by definition. Since Γ_{xmin} is strongly regular, any resources that contain machines cannot be used to generate those machines, by definition. This implies that machines additional to x are not needed. Thus, the set $S_x = \{x\} \cup (R_{xmin} \backslash M_x)$ is a valid seed.

Therefore, S_x is a seed for all Γ_x.

□

Proposition 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. Proposition 5 shows that the output of the algorithm is a seed.

Thus, completeness is guaranteed. □

Proposition 7.

Proof. Note that each iteration of the algorithm removes elements from a set with finite cardinality, and the algorithm stops once the set is depleted.

Consider the time complexity of Steps 1 to 5 during the first iteration of the algorithm.

In Step 1, the use of either one of the BFS or DFS algorithms has time complexity $O(n + m)$ [13].

In Step 2, 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 $O(n)$.

In Step 3, the time complexity of the DMST algorithm is $O(n_p m_p)$ [15], where n_p is the number of machines in the primary subsystem, and m_p is the number of resources in the primary subsystem. Accounting for the possibility that there is more than one matriarch to apply the DMST algorithm to, and that the cost of the seed for each matriarch's subsystem needs to be evaluated, the time complexity of this step is $O(n_{\text{♀}} n_p m_p + n_{\text{♀}})$, where $n_{\text{♀}}$ is the number of matriarchs.

In Step 4, 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 $O(m_p)$.

In Step 5, 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 of Steps 1 to 5 during the first iteration of the algorithm is $O(2n + m + n_p + m_p + n_{\text{♀}} n_p m_p + n_{\text{♀}})$.

In the best case, the SI algorithm executes once and there is only one matriarch. This implies that $n_p = n$, $m_p = m$ and $n_{\text{♀}} = 1$, so that the resultant best-case time complexity is $O(3n + nm + 2m + 1)$.

In the worst case, either the SI algorithm executes once and there are n matriarchs, or the SI algorithm executes n times and each machine is a matriarch for a family that has a singleton set of machines. Thus, we have two possibilities to consider.

The first possibility is $n_p = n$, $m_p = m$ and $n_{\text{♀}} = n$, so that the resultant time complexity is $O(n^2 m + 4n + 2m)$.

The second possibility is $n_p = 1$ and $m_p = 1$ implying that $n_{\text{♀}} = 1$, and after the first pass through the algorithm, Steps 2-5 are repeated $n - 1$ times. This time complexity is $O(2n + m + 4) + O((n - 1)(n + 4)) = O(n^2 + 5n + m)$. □

Proposition 8.

Proof. Let $\Gamma = (U, M, R, G)$ be made up of k strongly regular disjoint families. From Proposition 1, there are at least k matriarchs. Since each family is the generation subsystem of a matriarch, and each of these subsystems is strongly regular, Proposition 4 indicates that the minimum $|M_S|$ is k.

In the proof of Proposition 5, we have shown that each pass through Steps 1 to 4 of the SI algorithm produces a seed for a family, before the family is removed from the original generation system. This seed for the family contains one machine. If there are k 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. Thus, there will be k machines in M_S.

Therefore, the number of machines is optimal because it is the minimum it could be.

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. If there are multiple

matriarchs, the resource set that is selected is the least costly. Taking all such minimal cost resources produces an optimal seed resource set for each family, and since the families are disjoint, the union of these sets result in a seed that is optimal with respect to the cost of the resources. □

Proposition 9.

Proof. As a result of expanding M or R, the rank of a family will either increase or stay constant. This is because there are now more machines and resources in the generation system, and so it is possible that machines originally located in the outer layer are now able to produce offspring. Hence, it is possible that the rank increases.

Consider the original family, Γ, prior to the expansion of M or R. Let x be a machine in the outer layer of Γ. Since generation always proceeds outwards [5] and Γ is a family, expanding M or R may result in an increase in the rank of the system. If this occurs, there is now a degenerate machine y that is a descendant of x. In other words, now $\exists(r_n) \in R$ such that $G(x,(r_n)) = y$.

Since y is degenerate, it does not need to belong to M_S, so that $|M_S|$ remains the same. Also, if (r_n) uses resources that already belong to R_S, then $|R_S|$ stays unchanged.

However, if (r_n) uses resources that differ from those in R_S, then these resources need to be added to the resource seed set. Hence, $|R_S|$ increases, producing a corresponding increase in $|S|$. □

References

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

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

[3] M. Sipper, "Fifty years of research on self-replication: An overview," *Artifical Life*, vol. 4, no. 3, pp. 237–257, 1998.

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

[5] P. Kabamba, "The von neumann threshold of self-reproducing systems: Theory and computation," The University of Michigan, Tech. Rep. CGR-06-11, 2006.

[6] A. Menezes and P. Kabamba, "Information requirements for self-reproducing systems in lunar robotic colonies," in *Proceedings of the 57th International Astronautical Congress*, no. IAC-06-A5.P.04, 2-6 October 2006.

[7] ——, "A combined seed-identification and generation analysis algorithm for self-reproducing systems," in *Proceedings of the 2007 American Control Conference*, 11-13 July 2007, pp. 2582–2587.

[8] G. S. Chirikjian, Y. Zhou, and J. Suthakorn, "Self-replicating robots for lunar development," *IEEE/ASME Transactions on Mechatronics*, vol. 7, no. 4, Dec. 2002.

[9] J. Suthakorn, Y. T. Kwon, and G. S. Chirikjian, "A semi-autonomous replicating robotic system," in *Proceedings of the 2003 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, Jul. 2003.

[10] B. Foing, "Roadmap for robotic and human exploration of the moon and beyond," in *Proceedings of the 56th International Astronautical Congress*, no. IAC-05-A5.1.01, 17-21 October 2005.

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

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

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

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

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

[16] J. Hopcroft and R. Tarjan, "Algorithm 447: Efficient algorithms for graph manipulation," *Communications of the ACM*, vol. 16, no. 6, pp. 372–378, 1973.

[17] Y. J. Chu and T. H. Liu, "On the shortest arborescence of a directed graph," *Scientia Sinica*, vol. 14, pp. 1396–1400, 1965.

[18] J. Edmonds, "Optimum branchings," *Journal of Research of the National Bureau of Standards*, vol. 71B, no. 4, pp. 233–240, 1967.

[19] F. Bock, "An algorithm to construct a minimum directed spanning tree in a directed network," *Developments in Operations Research*, pp. 29–44, 1971.

[20] R. M. Karp, "A simple derivation of Edmonds' algorithm for optimum branchings," *Networks*, vol. 1, pp. 265–272, 1971.

[21] R. E. Tarjan, "Finding optimum branchings," *Networks*, vol. 7, pp. 25–35, 1977.