25th International Congress on Modelling and Simulation, Darwin, NT, Australia, 9 to 14 July 2023
mssanz.org.au/modsim2023

A simplified model for plant breeding

K. Davis [a], P. Le Bodic [a], A.T. Ernst [b], R. Kapoor [c], R. García-Flores [c]

[a] *Faculty of IT, Monash University, VIC 3800, Australia*
[b] *School of Mathematics, Monash University, VIC 3800, Australia*
[c] *CSIRO Data61, Private Bag 10, Clayton South, VIC 3169, Australia*
Email: Kelvin.Davis1@monash.edu

Abstract: This paper introduces a simplified deterministic model for plant breeding and an efficient algorithm to solve it. The model abstracts away much of the complexity of plant breeding to a version that can be solved in polynomial time. While this model only considers single-point recombinations and assumes all desired crossings are successful, it provides a lower bound on the number of generations required to achieve the target. The algorithm presented exploits runs of favourable alleles on producible gametes to guide its decision-making, which highlights the importance of considering segments rather than individual alleles. This paper suggests that seg-ments will be instrumental in the development of efficient algorithms for more comprehensive plant breeding models. However, future research should explore extending the model to account for factors such as polyploid plants and resource constraints that breeders face in reality. Overall, this paper serves as the first in a series of models exploring dedicated solving techniques applicable to plant breeding problems.

Keywords: *Plant breeding, crossing schedule, ideotype, segments, dominance*

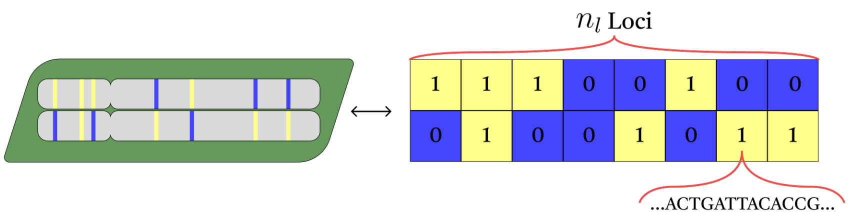

Figure 1. An example genotype with a single pair of chromosomes and $n_l = 8$ loci represented by a $2 \times n_l$ bit matrix. Yellow cells (1) represent favourable alleles and blue cells (0) represent unfavourable alleles.

1 Introduction

Plant breeding is the process of improving the traits of agricultural crops via the cross-breeding (*crossing*) of parent plants. Typically, this starts by mapping the locations on the plants genome (referred to as *loci*) that control the trait of interest and measuring the effect of each *allele* (the substring of DNA encoded at each locus) on that trait. Using this mapping, an ideal target plant can be specified and a *crossing schedule* that prescribes a partially ordered set of crossings to create the target (El-kebir 2009), starting from an initial population of plants, can be derived. Given the time and resource intensive nature of breeding programs, breeders aim to create crossing schedules that minimise the number of generations, crossings, and resources required to create the target.

Over the last two decades, the planning of plant breeding programs has been increasingly modelled using operations research tools. Individuals (plants) are often modelled as vectors or matrices of bits, with a 1 representing if an allele is favourable and 0 otherwise (Moeinizade et al. 2021), as shown in Figure 1. In this fashion, a population of plants can be encoded as a binary input to an optimisation problem. Furthermore, the children of any pair of parents must be a combination of bits sourced from either parent. Altogether, this allows plant breeding to be modelled as the creation of an individual with only favourable genes (i.e. bit vectors or matrices of 1) from a starting population under some objective.

Several approaches have been explored to solve plant breeding problems optimally. Such approaches include enumeration (Servin et al. 2004), dynamic programming (El-kebir 2009), and Mixed Integer Programming (Canzar & El-Kebir 2011, Xu et al. 2011, De Beukelaer et al. 2015). However, these approaches are limited in instance sizes that can be solved as they scale exponentially with respect to their input, with Xu et al. solving under 50 loci instances at max. In practice, some plant breeding problems involve thousands of locations on the genome. As such, current optimal solving techniques are ill-equipped to deal with problems of that size. Heuristic and Monte Carlo approaches have also been employed to tackle these problems as well (Meuwissen et al. 2001, Daetwyler et al. 2015, Han et al. 2017, Moeinizade et al. 2021) but these methods lose the benefit of optimality. So far little work has been done to isolate the components of the problem that can be solved efficiently as subproblems.

In this paper, we present a vastly simplified model of plant breeding that minimises the number of generations required to create a given target under a highly restrictive set of assumptions. Such a model will serve as the backbone to a hierarchy of models presented in future works exploring the effect of different complicating components, eventually coalescing in many of the practical models addressed in the literature and improved algorithms to solve them. We derive an efficient algorithm that makes greedy choices over the runs of consecutive favourable alleles and contrast its efficiency against two brute force algorithms that ignore these runs.

The rest of the paper is laid out as follows. Section 2, establishes the assumptions made, notation used, and the decision problem solved in this paper. Section 3 describes the preconditions for feasibility before detailing an efficient algorithm to solve the problem. Section 4 presents the computational experiments used to highlight the efficiency of the algorithm presented in Section 3 . Finally, the ramifications and limitations of the work are presented in Section 5.

2 Problem Specification

In order to define an abstraction of the problem, we make the following assumptions: (1) any two individuals can cross, including with themselves, (2) all *recombinations* (crossovers) are single-point, (3) when crossing two individuals, we decide which offspring, among all possible offspring, are generated, (4) the alleles of all input and generated chromosomes and gametes are known, (5) mutations are not considered, (6) all individuals are diploid, i.e. they consist of two chromosomes, one from each parent.

The reasons for these assumptions are: (1) this corresponds to angiosperms, i.e. all plants that flower, which is most plants, (2) multi-point recombinations are rare, and harder to deal with algorithmically, (3) we assume that, in practice, in order to generate the desired offspring from any pair of parents, we would cross the parents sufficiently many times, (4) finding the genotype of an individual is generally possible at a reasonable cost through the use of *markers*, (5) undesirable mutations can be avoided for the reasons given for (3), and desirable mutations are too random to plan around, (6) while many plants are polyploid, we start with diploid as it is the simplest to solve.

With these assumptions, there is no uncertainty left either in the input or in the crossings that could be used within a breeding schedule. Therefore, what remains of the problem is a deterministic optimisation problem that consists in minimising the number of generations required to produce the target individual.

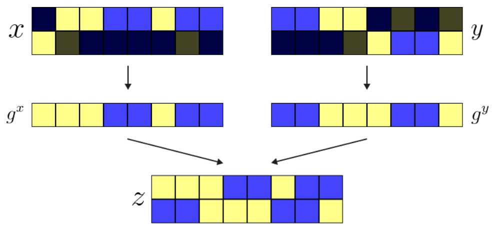

Figure 2. An example of a crossing between individuals x and y to produce z. Each individual performs recom-bination to produce gametes g^x and g^y before these gametes fuse to produce z. Contrastive highlighting is used to show which alleles are used to source g^x and g^y, for instance, g^y is sourced from prefix $y_{1,1...4}$ and suffix $y_{2,5...8}$.

We refer to individuals by their *genotype*, a specification of the alleles at each locus on each chromosome. Formally, we represent diploid genotypes as pairs of bit strings, i.e. $x \in \{0, 1\}^{2\times n_l}$ where n_l is the number of loci on the chromosome and each bit $x_{i,j}$ represents the allele on chromosome i at locus j. The target individual x^* is thus a genotype whose alleles are all favourable, i.e. $x^*_{i,j} = 1 \ \forall i \in \{1, 2\} \ j \in [n_l]$. We denote $\mathcal{P}_t$ as the population of genotypes at generation t with $\mathcal{P}_0$ being the initial population.

The crossing of individuals involves the fusion of pollen and egg cells, referred to as *gametes*, from each parent. In our model, a gamete $g^x \in \{0, 1\}^{n_l}$ of an individual x, is created by sourcing bits from either $x_{1,p}$ or $x_{2,p}$ for all loci $p \in [n_l]$ such that the number of times the source chromosome switches from x_1 to x_2 or visa versa is at most 1. Thus, crossing individuals x and y to create z involves creating gametes g^x and g^y and fusing them together to create the offspring $z = (g^x, g^y)$ as shown in Figure 2. We define g^* as a gamete with all favourable alleles $g^* = (1, \ldots, 1)$ such that $x^* = (g^*, g^*)$. We denote $\mathcal{X}_t$ as the set of gametes producible by any genotype in $\mathcal{P}_t$, i.e. $\mathcal{X}_t = \{g^x \mid g^x \in X_x, \ x \in \mathcal{P}_t\}$ where X_x is the set of gametes producible from x. Furthermore, let $G_{x,y}$ be the set of offspring of parents x and y.

Definition 2.1 (Crossing Schedule)**.** Inspired by the definition provided by Canzar & El-Kebir (2011), for any set X, initial subset $X_0 \subset X$, predicate $R(x, y, z)$ and target $x^* \in X$, let a crossing schedule T over (X, X_0, R, x^*) be a directed acyclic graph from source nodes containing members of X_0 to a single sink node containing x^*. All internal nodes of T, including the sink node, have exactly two parents where for all internal node z with parents x and y, the relation $R(x, y, z)$ holds. Furthermore, the number of generations required to produce any x is its depth from the source nodes. Thus, the number of generations required to produce x^* is given by the height of T.

It is worth noting that $R(x, y, z)$, introduced here to parameterise crossing schedules for different classes of objects, is a relation that constrains the domain of z subject to x and y. In the problem addressed in this paper, $R(x, y, z)$ holds if and only if z is one of any of the possible offspring of parents x and y, i.e. $z \in G_{x,y}$. Thus, implicit to finding any crossing schedule T is the realisation, for each internal node z of T, of the recombinations required to create z from its parents in order for R to be satisfied. We henceforth refer to this instance of R as mateGen(x, y, z).

2.1 Decision Problems

We define the following decision problem to be answered in this paper.

Problem 2.2 (CROSSINGSCHEDULE)
Input: initial population of genotypes $\mathcal{P}_0$ and generation number $t_{\max}$.
Question: Does a crossing schedule T over $(\mathcal{P}, \mathcal{P}_0, \text{mateGen}, x^*)$ exist with height $\leq t_{\max}$?

In order to solve Problem 2.2, we perform a chain of reductions to simplify the problem. For any non-trivial problem where $x^* \notin \mathcal{P}_0$, creating x^* requires creating two copies of g^* starting from the population of gametes $\mathcal{X}_0$ producible from $\mathcal{P}_0$. New gametes can be created through the process of fusing parent gametes into and producing new gametes from the intermediate individual, $g^z \in X_{(g^x, g^y)}$. Let $\text{mateGam}(g^x, g^y, g^z)$ be the predicate that describe this relation. The following decision problem encapsulates this transformation and is paired with a reduction from Problem 2.2 below.

Problem 2.3 (CROSSINGSCHEDULEGAMETE)
Input: initial population $\mathcal{X}_0$ of gametes and generation number $t'_{\max}$.
Question: Does a crossing schedule T' over $(\mathcal{X}, \mathcal{X}_0, \text{mateGam}, g^*)$ exist with height $\leq t'_{\max}$?

Reduction from Problem 2.2 to Problem 2.3. Given an instance $(\mathcal{P}_0, x^*, t_{\max})$ of Problem 2.2, instance $(\mathcal{X}_0, g^*, t'_{\max})$ of Problem 2.3 is constructed by letting $\mathcal{X}_0$ be an initial population that contains all of the gametes that can be produced by any genotype in $\mathcal{P}_0$, i.e. $\mathcal{X}_0 = \{g^x \mid g^x \in X_x,\ x \in \mathcal{P}_0\}$ the new target g^* is defined as $g^* = (1, \ldots, 1)$ where $g^* \in \{0,1\}^{n_l}$, and $t'_{\max} = \max(0, t_{\max} - 1)$ as an extra generation is required to create x^* from its constituent g^* gametes.

The next problem describes the flow of favourable genetic material from $\mathcal{X}_0$ to g^* and highlights which runs of favourable alleles (referred to as *segments*) in each gamete contribute to g^*.

Definition 2.4 (segment)**.** A favourable contiguous segment (abbrev. segment) is a tuple $c_x = (s_x, e_x, g^x)$ that represents a maximal contiguous sequence of indices from s_x to e_x where $\forall p \in \{s_x \ldots e_x\},\ g^x_p = 1$. Note that for any segment (s_x, e_x, g^x), $s_x \leq e_x$.

Let $\mathcal{S}_t$ be the set of all segments on gametes in $\mathcal{X}_t$. The goal of Problem 2.3 is to create g^* which both contains and requires the creation of a $c^* = (1, n_l, g^*)$ segment. Thus the number of generations required to create g^* is also the number of generations required to create c^*. Any two overlapping or adjacent segments, i.e. segments $c_x = (s_x, e_x, g^x)$ and $c_y = (s_y, e_y, g^y)$ where $s_x < s_y \leq e_x + 1 \leq e_y$, can be joined via recombination to create a segment that covers the union of c_x and c_y, namely $c_z = (s_x, e_y, g^z)$, where $g^z \in X_{(g^x, g^y)}$ is a gamete created via recombination in the intersection of c_x and c_y. Similar to mateGen and mateGam, $\text{mateSeg}(c_x, c_y, c_z)$ describes this relation. Segment pairs that do not meet this condition are either disjoint and so cannot be joined into a single segment in a single recombination, or are not worth joining because one of the segments is covered by the other. Thus, the second reduction transforms Problem 2.3 into the following decision problem:

Problem 2.5 (CROSSINGSCHEDULE)
Input: initial population $\mathcal{S}_0$ and generation number $t''_{\max}$.
Question: Does a crossing schedule T'' over $(\mathcal{S}, \mathcal{S}_0, \text{mateSeg}, c^*)$ exist with height $\leq t''_{\max}$?

Reduction 2 from Problem 2.3 to Problem 2.5. Given an instance $(\mathcal{X}_0, g^*, t'_{\max})$ of Problem 2.3, the instance $(\mathcal{S}_0, c^*, t''_{\max})$ of Problem 2.5 is constructed by letting $\mathcal{S}_0$ be set of all segments attached to any gamete in $\mathcal{X}_0$, $c^* = (1, n_l, g^*)$, and $t''_{\max} = t'_{\max}$ as g^* implies the existence of c^* and visa versa.

3 METHODS

3.1 Feasibility

Here we present the feasibility conditions for Problem 2.5. An instance of Problem 2.5 is feasible if and only if for every locus $p \in [n_l]$ there is at least one segment $c \in \mathcal{S}_0$ that contains it, i.e. $\forall p \in [n_l],\ \exists c \in \mathcal{S}_0,\ p \in c$. This condition is necessary as every allele in any segment c_z and, by extension, c^* must be sourced from one of its ancestors in $\mathcal{S}_0$. Assuming this condition holds, we show that this is also sufficient for feasibility by way of an algorithm to solve Problem 2.5.

3.2 Segment-based algorithm

We now derive an algorithm that constructs an optimal crossing schedule from $\mathcal{P}_0$. In the special case where $x^* \in \mathcal{P}_0$, The optimal crossing schedule T consists of a single node representing x^* whose height is 0. The remainder of this section assumes $x^* \notin \mathcal{P}_0$ and feasibility holds. After extracting $\mathcal{S}_0$ from $\mathcal{P}_0$, all that remains is to form minimum height crossing schedule T'' from segments in $\mathcal{S}_0$ to c^*.

Let $\mathcal{S}^* = \{c_1, \ldots, c_{n_s}\}$ be a minimal subset of segments of $\mathcal{S}_0$ that covers $[n_l]$ ordered by start point. All successive pairs of segments $c_i, c_{i+1} \in \mathcal{S}^*$ can be joined in the manner described above. As these joins are associative, they can be arranged as the internal nodes of any binary tree whose leaves from left to right correspond to segments in $\mathcal{S}^*$. The shallowest such tree is a balanced binary tree with height $\lceil \log_2(|\mathcal{S}^*|) \rceil$. Thus, the algorithm to answer Problem 2.2 works by extracting all segments from $\mathcal{P}_0$ and choosing the minimum covering subset $\mathcal{S}_0$. The optimal crossing schedule is produced by joining the chosen segments in pair-wise fashion until c^* is created, as shown in Figure 3.

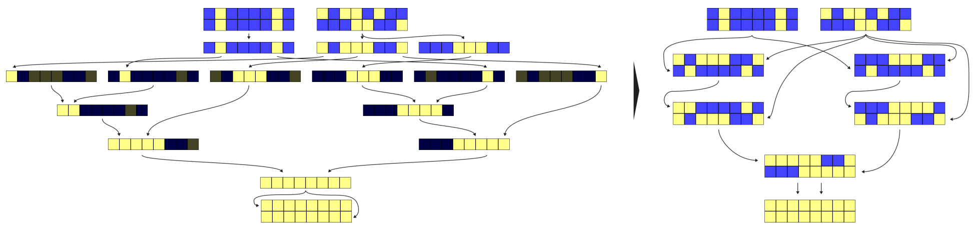

Figure 3. An illustration of a crossing schedule constructed using the segment-based algorithm with (left) the extraction and joining of $\mathcal{S}^*$ from $\mathcal{P}_0$, followed by (right) the crossing schedule created using these joins under the mappings in Appendix A. While this figure depicts an instance where $\mathcal{P}_0$ contains only two individuals, instances of Problem 2.2 can contain any non-negative number of individuals in $\mathcal{P}_0$.

Using the results derived, Problem 2.2 can be answered in $\mathcal{O}(n_l|\mathcal{P}_0|)$ and its crossing schedule can be found in $\mathcal{O}(n_l \cdot (|\mathcal{P}_0| + n_l))$. Extraction of $\mathcal{S}^*$ can be achieved in $\mathcal{O}(n_l|\mathcal{P}_0|)$ time by maintaining an array of size n_l that stores the largest segment with start point s_x found so far at index s_x of the array and performing at a single pass over the array to select the minimum covering subset. The cardinality of $\mathcal{S}^*$ is bounded by n_l as each segment must cover at least one unique locus and $\mathcal{P}^*$ must cover $[n_l]$. The cost of joining of any two segments costs $\mathcal{O}(n_l)$, therefor the construction the crossing schedule from $\mathcal{S}^*$ has a time complexity of $\mathcal{O}(n_l^2)$.

4 Results

We benchmark the segment-based algorithm derived in Section 3 against two naive algorithms. The first of these algorithms repeatedly breeds all possible offspring from all possible pairs of parents until x^* is found. The second algorithm we benchmark against improves on the first by removing all genotypes dominated by other genotypes in each generation where any x is dominated by y if every favourable allele in x is also in y.

The three algorithms were tested on uniform random feasible instances whose initial populations $\mathcal{P}_0$ contain $|\mathcal{P}_0| = 6$ individuals each with n_l loci varying from $n_l = 1$ to $n_l = 10^4$. This allows testing of scenarios where the segments in $\mathcal{S}^*$ can be drawn from a single to multiple genotypes without excessive computational cost for the naive algorithms. For each benchmark, 1000 instances are sampled for each value of n_l and the average runtime is recorded. The algorithms are implemented in Python 3.7.11 and are run single-threaded on an Intel i7-8750H processor running at 4.100GHz.

Figure 4 presents the average runtime as a function of the number of loci in the initial population. The results clearly demonstrate the superiority of the segment-based approach over the naive methods. Notably, the runtime of the naive algorithms exhibits a significant exponential increase, with the first and second algorithms taking more than 1 second to complete at 10 and 16 loci, respectively. In contrast, the segment-based algorithm can efficiently solve much larger instances, handling up to 10,000 loci instances in less than 0.1 seconds. Overall, these findings highlight the potential efficiency benefits of segment-based approaches to plant breeding.

5 Discussion and Conclusion

This paper presented a simplified deterministic model for plant breeding and an efficient algorithm to solve it. Such a model while not immediately usable to plant breeders, will serve as the first in a series of models exploring solving techniques applicable to plant breeding problems. The model presented abstracts away much of the complexity of plant breeding to a version that can be solved in polynomial time. Given that this model assumes the success of each crossing in the optimal schedule, solving this model provides a lower bound on the number of generations that would be required to achieve the target in practice, as well a crossing schedule that can be achieved even with great cost. This implies that the algorithm for this model can be used as a heuristic in algorithms, such as A*, to solve more complex variants. Finally, simplifying the model to this extent led to an algorithm that utilised the runs of favourable alleles on producible gametes to guide its decision making.

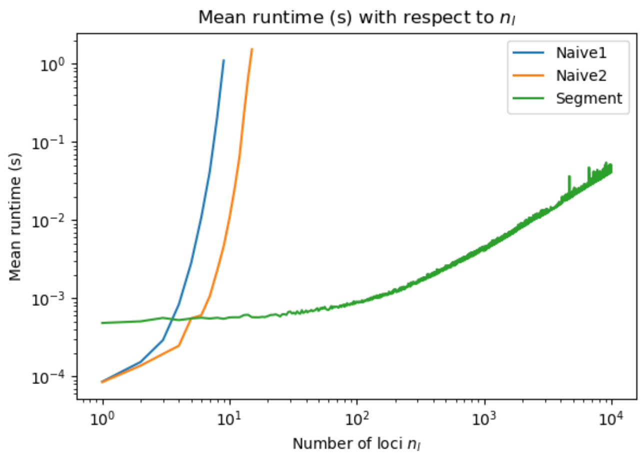

Figure 4. A log-log plot of the average runtime in seconds of the two naive algorithms (blue and orange) and the segment-based algorithm (green) with respect to the number of loci n_l. Each data point is the average runtime over 1000 random feasible instances.

Core to the efficiency of this algorithm is the exploitation of segments, whose utility has remained largely unexplored in previous works to date. The algorithm presented highlights the importance of paying consideration to the runs of favourable alleles rather than the individual alleles by themselves. This is simply an extension of alleles which, on their own, are already abstractions over the nucleotide bases that constitute the individual's DNA. We posit that segments will be instrumental in the development of efficient algorithms for more comprehensive plant breeding models.

While these assumptions simplify the problem, they may not hold in practice, especially for plants with multiple sets of chromosomes or for breeders with limited resources. Thus, future research should explore how to extend our model to account for these factors and develop solving techniques that can handle these extensions. Furthermore, while our model and algorithm have shown promising results, there are still several open questions that need to be addressed. For instance, we have not explored how our approach would perform for polyploid plants, which are common in many agricultural crops. Moreover, we have not considered the resource constraints that breeders face in reality, which could limit the applicability of our algorithm. Future research should investigate these issues and develop more comprehensive models and algorithms that can handle these challenges.

References

Canzar, S. & El-Kebir, M. (2011), A mathematical programming approach to marker-assisted gene pyramiding, *in* T. M. Przytycka & M.-F. Sagot, eds, 'Algorithms in Bioinformatics', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 26–38.

Daetwyler, H. D., Hayden, M. J., Spangenberg, G. C. & Hayes, B. J. (2015), 'Selection on optimal haploid value increases genetic gain and preserves more genetic diversity relative to genomic selection', *Genetics* **200**(4), 1341–1348.

De Beukelaer, H., De Meyer, G. & Fack, V. (2015), 'Heuristic exploitation of genetic structure in marker-assisted gene pyramiding problems', *BMC Genetics* **16**(1), 16.

El-kebir, M. (2009), Crossing schedule optimization, Master's thesis, Eindhoven University of Technology.

Han, Y., Cameron, J. N., Wang, L. & Beavis, W. D. (2017), 'The predicted cross value for genetic introgression of multiple alleles', *Genetics* **205**(4), 1409–1423.

Meuwissen, T. H., Hayes, B. J. & Goddard, M. E. (2001), 'Prediction of total genetic value using genome-wide dense marker maps', *Genetics* **157**(4), 1819–1829.

Moeinizade, S., Han, Y., Pham, H., Hu, G. & Wang, L. (2021), 'A look-ahead Monte Carlo simulation method for improving parental selection in trait introgression', *Scientific Reports* **11**(1), 1–12.

Servin, B., Martin, O. C., Mézard, M. & Hospital, F. (2004), 'Toward a theory of marker-assisted gene pyramiding', *Genetics* **168**(1), 513–523.

Xu, P., Wang, L. & Beavis, W. D. (2011), 'An optimization approach to gene stacking', *European Journal of Operational Research* **214**(1), 168–178.

A Reduction proofs

Let T, T', and T'' be the optimal crossing schedules that answer Problem 2.2, Problem 2.3, and Problem 2.5, respectively.

A.1 Proof of Reduction 1

Let the mapping T' to T, performed by function $f_{T' \to T}$, be defined as follows:

- For all source nodes in T' representing g^x, create a source node in T representing the unique genotype $x \in \mathcal{P}_0$ that g^x was sourced from.
- For all gametes g^z in T' with parents g^x and g^y, there is a node in T representing z with parents x and y where $z = (g^x, g^y)$; for the triple of nodes in T representing x, y, and z, edges $\langle x, z \rangle$ and $\langle y, z \rangle$ are created in T.
- A sink node is created for x^* and a pair of edges from the current sink node this new node representing x^*. This represents the final selfing of T.

Under this mapping, any such T mapped from T' has $height(T) = height(T') + 1$ as the number of generations to produce x^* is the number of generations to produce g^* with one more for x^* to mature.

Lemma A.1. *If T' has minimum height then $f_{T' \to T}(T')$ also has minimum height.*

Proof. If T' has minimum height, then the minimum number of generations to create g^* is $height(T')$. $x^* = (g^*, g^*)$ requires two g^* gametes and thus, assuming $x^* \notin \mathcal{P}_0$, the minimum number of generations to create x^* is $height(T') + 1$. $f_{T' \to T}(T')$ also has height $height(T') + 1$, so $f_{T' \to T}(T')$ must therefore have minimum height. $\square$

Using the mapping $T' \to T$, the Yes and No answers are preserved by the reduction.

A.2 Proof of Reduction 2

Let the mapping T'' to T' be defined as follows:

- For each node in T'' representing segment (s_x, e_x, g^x), create a node in T' representing g^x with only one node for each unique g^x.
- For each edge $\langle (s_x, e_x, g^x), (s_z, e_z, g^z) \rangle$ in T'', create an edge $\langle g^x, g^z \rangle$ in T'.

Under this mapping, any such T' mapped from T'' has $height(T') = height(T'')$ as creating c^* implies creating g^* and visa versa. Thus, the Yes and No answers are preserved by the reduction from Problem 2.3 to Problem 2.5.