Web Appendices for A Divide and Conquer Algorithm for Exploiting Policy Function Monotonicity

Grey Gordon and Shi Qiu*

October 10, 2017

Abstract

Appendix C gives additional sufficient conditions for monotonicity and shows how the results may be applied. Appendix D shows the algorithms deliver correct solutions and incidentally shows how additional state variables (for which one does not want to exploit monotonicity) may be handled. Appendix E quantitatively evaluates binary monotonicity with sorting while also showing (1) how binary monotonicity with sorting can be used to solve some problems that do not immediately fit the class of non-monotone problems and (2) how choice variables for which one does not want to (or cannot) exploit monotonicity may be handled. Appendix F contains all the omitted proofs and lemmas.

C Additional sufficient conditions and applications

This appendix expands on the discussion of monotonicity sufficient conditions in Section 5 in two ways. First, Section C.1 gives sufficient conditions for choice correspondences to be ascending and for functions to exhibit increasing differences. Second, Section C.2 applies these sufficient conditions and the ones from the main text to establish monotonicity in the RBC model, the Arellano (2008) model, and the sorted problem (6). Appendix F gives the proofs.

C.1 Ascending correspondences and increasing differences

Proposition 3 requires the choice correspondence to be ascending. One way to ensure this is for every choice to be feasible. The following lemma, which relies on increasing differences, provides an alternative.

Lemma 1. *Let $\mathcal{I}, \mathcal{I}' \subset \mathbb{R}$ and $I' : \mathcal{I} \to P(\mathcal{I}')$. Suppose $I'(i) = \{i' \in \mathcal{I}' | h_m(i, i') \geq 0$ for all $m \in M\}$ with M arbitrary. If I' is increasing, h_m is decreasing in i', and h_m has increasing differences on $\mathcal{I} \times \mathcal{I}'$ (for all m), then I' is ascending on $\mathcal{I}$.*

*Grey Gordon (corresponding author), Indiana University, Department of Economics, greygordon@gmail.com. Shi Qiu, Indiana University, Department of Economics, shiqiu@indiana.edu.

For an example application of Lemma 1, consider the RBC model and define $c(k', k, z) := -k' + zF(k) + (1-\delta)k$. Since c is increasing in k and decreasing in k', the budget constraint $\{k' \in \mathcal{K} | c(k, k', z) \geq 0\}$ will be ascending on $\mathcal{K}$ for each z as long as c has increasing differences in k, k'.

To apply the sufficient conditions Proposition 3 and Proposition 4 or to apply Lemma 1, one must establish functions have increasing differences. The following lemma provides a number of sufficient conditions for establishing that this is the case. Some additional conditions may be found in Topkis (1978) and Simchi-Levi, Chen, and Bramel (2014).

Lemma 2. *For $\mathcal{I}, \mathcal{I}' \subset \mathbb{R}$ and $S \subset \mathcal{I} \times \mathcal{I}'$, $f : S \to \mathbb{R}$ has increasing differences on S if any of the following hold:*

(a) $f(i, i') = p(i) + q(i')$ for arbitrary p and q.

(b) $f(i, i') = p(i) + q(i') + r(i)s(i')$ for arbitrary p and q with r and s both increasing or decreasing.

(c) $f(i, i')$ agrees with $g : L \subset \mathbb{R}^2 \to \mathbb{R}$, a C^2 function having $g_{12} \geq 0$ and L a hypercube with $S \subset L$.

(d) $f(i, i')$ is a non-negative linear combination (i.e., $f = \sum \alpha_k f_k$ with $\alpha_k \geq 0$) of functions having increasing differences.

(e) $f(i, i') = h(g(i, i'))$ for h an increasing, convex, C^2 function and g increasing (in i and i') and having increasing differences.

(f) $f(i, i') = h(g(i, i'))$ for h an increasing, concave, C^2 function and g increasing in i, decreasing in i', and having increasing differences.

(g) $f(i, i') = \int_E g(h(i, \varepsilon), i') dF(\varepsilon)$ with g having increasing differences on $\{(\hat{h}, i') | \hat{h} = h(i, \varepsilon), \varepsilon \in E, (i, i') \in S\}$ and h increasing in i.

(h) $f(i, i') = \max_{x \in \Gamma(i, i')} g(i, i', x)$ exists for all $(i, i') \in S$, S is a lattice, $\Gamma : S \to P(X)$, $X \subset \mathbb{R}$, the graph of Γ is a lattice, and g has increasing differences in i, i' and i, x and i', x on $\mathcal{I} \times \mathcal{I}'$, $\mathcal{I} \times X$, and $\mathcal{I}' \times X$, respectively (for all i, i', x).

Since our method primarily exploits monotonicity of one choice variable, it will be convenient in some cases to construct an indirect utility function over the other choices and then establish that the indirect utility function has increasing differences. Lemma 2 part (h) gives one sufficient condition for this, but its conditions can be difficult to guarantee unless every choice is feasible. Proposition 5 provides an alternative sufficient condition that may be easier to verify.

Proposition 5. *Let $S \subset \mathbb{R}^2$ be open and convex. Let $f : S \to \mathbb{R}$ be defined by $f(i, i') = \max_{x \in X} u(g(x, i, i'), x)$ where u is differentiable, increasing, and concave in its first argument and X is arbitrary. Then f has increasing differences on S if*

1. *f is well-defined and C^1 in i on the closure of S, and*
2. *for any optimal policy x^* and any $(i, i') \in S$, $g_2(x^*(i, i'), i, i')$ exists, is positive, and is increasing in i' and $g(x^*(i, i'), i, i')$ is decreasing in i'.*

C.2 Applying the sufficient conditions

We now show how the preceding results can be applied to establish monotonicity in the RBC model, Arellano (2008), and the sorted problem (6).

C.2.1 Monotonicity in k in the RBC model

Consider the RBC model where we defined $c(k', k, z) = -k' + zF(k) + (1-\delta)k$. By Lemma 2 part (a), c has increasing differences in k, k'. Then, $u(c(k', k, z))$ and $u(c(k', k, z)) + \beta \mathbb{E}_{z'|z} V_0(k', z')$ have increasing differences by parts (f) and (d), respectively. Consequently, an application of Proposition 3 (noting the budget constraint is ascending by Lemma 1) gives that $\arg\max_{k'} u(c(k', k, z)) + \beta \mathbb{E}_{z'|z} V_0(k', z')$ is ascending on $\mathcal{K}$. As stated in the main text and proven in Appendix D, this means binary monotonicity can be used to compute an optimal policy $k'(k, z)$ that is monotone in k.

C.2.2 Monotonicity in z in the RBC model

With additional assumptions, one can also use these results to establish k' is monotone in z, although doing so is more complicated. Identical arguments to the above give the optimal choice correspondence as ascending in z if $\mathbb{E}_{z'|z} V_0(k', z')$ has increasing differences in k', z. By Lemma 2 part (g), this will hold as long as z' is increasing in z and $V_0(k, z)$ has increasing differences in k, z. To ensure this is the case at each step of the Bellman update (assuming the initial guess has increasing differences), one can use Lemma 2 part (h) if the graph of the choice correspondence is a lattice and $u \circ c + \beta \mathbb{E}_{z'|z} V_0$ has increasing differences in k, z. A sufficient condition for the former is that every choice is feasible. A sufficient condition for the latter, by Lemma 2 part (c), is that $\partial^2 u(c)/(\partial k \partial z) \geq 0$, which is the same condition required in Hopenhayn and Prescott (1992).

C.2.3 Monotonicity in k in the RBC model with elastic labor supply

One can establish monotonicity of $k'(k, z)$ in k for the RBC model with elastic labor supply by using Proposition 5 to establish increasing differences of the *indirect* utility function followed by an application of Proposition 3. Specifically, the maximization problem for a given z can be written as $\max_{k'} U(k, k') + \beta \mathbb{E}_{z'|z} V(k', z')$ where $U(k, k') := \max_{l \in [0,1]} u(c(l, k', k), l)$ and $c(l, k, k') := \max\{0, -k' + zF(k, l) + (1-\delta)k\}$. If U is differentiable and solutions are interior—i.e., $l^* \in (0, 1)$ and $c(l^*, k, k') > 0$—then $c_2(l^*, k, k') = zF_k(k, l^*) + 1 - \delta$ exists, is positive, and is weakly increasing in k'. If in addition consumption is a normal good so that $c(l^*, k, k')$ is decreasing in k', then Proposition 5 gives U as having increasing differences. In this case, Proposition 3 gives monotonicity of $k'(k, z)$ in k.

C.2.4 Monotonicity in the Arellano (2008) model and the sorted problem

Proposition 4 may be used to establish monotonicity in the Arellano (2008) model and the sorted problem (6). For the Arellano (2008) model, the budget constraint may be written as $c(b, b'; y) =$

$b + y - q(b', y)b'$, which is increasing in b. By Lemma 2 part (a), c has increasing differences in b, b'. Moreover, the continuation utility $W(b'; y) := \beta \mathbb{E}_{y'|y} V(b', y)$ (where V is the upper envelope of the repayment and default value functions) is weakly increasing in b'. Consequently, Proposition 4 applies. An identical argument may be used for the sorted problem.

D Algorithm correctness for a more general problem

This appendix shows binary monotonicity and the other algorithms deliver correct solutions in a more general formulation of (1). Section D.1 gives the more general formulation and states the correctness result. Section D.2 shows how the result applies in the RBC and Arellano (2008) models while incidentally showing how additional state variables, for which one does not want to exploit monotonicity, may be handled.

D.1 A more general formulation

In both the RBC and Arellano (2008) model, there is no guarantee that every choice is feasible. Consequently, one cannot directly use binary monotonicity because the maximization problems do not directly fit (1). To handle this issue in a general way, suppose that the feasible choice set is $I'(i) \subset \{1, \ldots, n'\}$, which may be empty, and that the objective function is given by some $\tilde{\pi}(i, i')$ only defined for (i, i') such that $i' \in I'(i)$. For every i such that $I'(i)$ is nonempty, define

$$\tilde{\Pi}(i) = \max_{i' \in I'(i)} \tilde{\pi}(i, i'). \tag{7}$$

Let $\underline{\pi}$ denote a lower bound on $\tilde{\pi}$,[7] and define

$$\pi(i, i') = \begin{cases} \tilde{\pi}(i, i') & \text{if } I'(i) \neq \emptyset \text{ and } i' \in I'(i) \\ \underline{\pi} & \text{if } I'(i) \neq \emptyset \text{ and } i' \notin I'(i) \\ \mathbf{1}[i' = 1] & \text{if } I'(i) = \emptyset, \end{cases} \tag{8}$$

Further, formalize the notion of concavity in the following way:

Definition 3. *The* problem is concave *if, for all i such that $I'(i) \neq \emptyset$, $I'(i) = \{1, \ldots, \overline{n}'(i)\}$ for some monotone increasing function $\overline{n}'(i)$ and $\pi(i, \cdot)$ is either first strictly increasing and then weakly decreasing; or is always weakly decreasing; or is always strictly increasing (where defined).*

Now we can state the correctness result:

Proposition 6. *If I' is increasing, then any of brute force, simple, or binary monotonicity combined with any of brute force, simple, or binary concavity applied to (1) with the objective function defined*

[7] A theoretical lower bound on $\tilde{\pi}$ is $-1 + \min_i \min_{i' \in I'(i)} \tilde{\pi}(i, i')$. While this particular bound is not practically useful (because computing it would be very costly), the smallest machine-representable number serves as a lower bound for computational purposes.

as in (8) delivers an optimal solution to (7) provided $\arg\max_{i' \in I'(i)} \tilde{\pi}(i, i')$ *is ascending and the problem is concave as required by the algorithm choices.*

For the proof, see Appendix F.

D.2 Examples

To see how this result may be applied and how the RBC model can be cast into (7), consider the problem's Bellman update,

$$\begin{aligned} V(k, z) = \max_{c \geq 0, k' \in \mathcal{K}} u(c) + \beta \mathbb{E}_{z'|z} V_0(k', z') \\ \text{s.t. } c + k' = zF(k) + (1 - \delta)k \end{aligned} \tag{9}$$

for $k \in \mathcal{K}$, $z \in \mathcal{Z}$ where $\mathcal{K} = \{k_1, \ldots, k_n\}$ with the k_i increasing. (While here we have used inelastic labor supply, elastic labor supply can be incorporated by replacing the period utility function with an an indirect utility function as we discuss in Appendix C.) Now, create a separate problem for each z and write

$$\tilde{\Pi}_z(i) = \max_{i' \in I'_z(i)} \tilde{\pi}_z(i, i') \tag{10}$$

where $\tilde{\pi}_z$ and I'_z are defined as

$$\begin{aligned} \tilde{\pi}_z(i, i') &:= u(-k_{i'} + zF(k_i) + (1 - \delta)k_i) + \beta \mathbb{E}_{z'|z} V_0(k_{i'}, z') \\ I'_z(i) &:= \{i' \in \{1, \ldots, n\} | k_{i'} \leq zF(k_i) + (1 - \delta)k_i\}. \end{aligned} \tag{11}$$

Then (10) is just (9) with k and k' given by grid indices. Moreover, (10) has the same form as (7). Further, because I'_z has the form $\{1, \ldots, \bar{n}'_z(i)\}$ for an increasing function $\bar{n}'_z(i)$, Proposition 6 shows the monotonicity and concavity algorithms will deliver correct solutions.

The Arellano (2008) model can be mapped into (7) in the same fashion. The main computational difficulty in that model is solving the sovereign's problem conditional on not defaulting. Specifically, the problem is to solve, for each $b \in \mathcal{B}$ and $y \in \mathcal{Y}$,

$$\begin{aligned} V^n(b, y) = \max_{c \geq 0, b' \in \mathcal{B}} u(c) + \beta \mathbb{E}_{y'|y} \max\{V^n(b', y'), V^d(y')\} \\ \text{s.t. } c + q(b', y)b' = b + y \end{aligned} \tag{12}$$

where b is the sovereign's outstanding bonds, $q(b', y)$ is the bond price, y is output, and V^d is the value of defaulting. Taking $\mathcal{B}$ as $\{b_1, \ldots, b_n\}$ with the b_i increasing and creating a separate problem for each y, one has

$$\tilde{\Pi}_y(i) = \max_{i' \in I'_y(i)} \tilde{\pi}_y(i, i') \tag{13}$$

where $\tilde{\pi}_y$ and I'_y are defined as

$$\begin{aligned}\tilde{\pi}_y(i,i') &:= u(-q(b_{i'},y)b_{i'}+b_i+y)+\beta\mathbb{E}_{y'|y}\max\{V^n(b_{i'},y'),V^d(y')\}\\ I'_y(i) &:= \{i'\in\{1,\ldots,n\}|-q(b_{i'},y)b_{i'}+b_i+y\geq 0\}.\end{aligned} \tag{14}$$

Then (13) has the same form as (7) and I'_y is increasing. Consequently, the monotonicity algorithms will deliver correct solutions.

E Additional results for the class of non-monotone problems

This appendix builds on Section 4 by testing the quantitative performance of binary monotonicity with sorting. The application, a sovereign default model with endogenous capital accumulation, is described in Section E.1. Section E.2 assesses the algorithm's performance inclusive of sorting costs. Last, Section E.3 shows how binary monotonicity with sorting can be used to solve some problems that do not fit (6) by transforming them into two-stage problems. It also illustrates how choice variables for which one does not want to (or cannot) exploit monotonicity may be handled.

E.1 A sovereign default model with capital

The model is similar to Bai and Zhang (2012) but lacks capital adjustment costs (in Section E.3 we use adjustment costs to illustrate how two-stage reformulations allow binary monotonicity to be used). A sovereign has total factor productivity a that is Markov and chooses bonds b' and capital k' from sets $\mathcal{B}$ and $\mathcal{K}$, respectively, with $0\in\mathcal{B}$. If the sovereign defaults, output ak^α falls by a fraction κ. In equilibrium, the discount bond price q satisfies $q(b',k',a)=(1+r)^{-1}\mathbb{E}_{a'|a}(1-d(b',k',a'))$ where r is an exogenous risk-free rate and d gives the default decision. The sovereign's problem is to solve

$$V(b,k,a)=\max_{d\in\{0,1\}} dV^d(k,a)+(1-d)V^n(b,k,a), \tag{15}$$

where the value of defaulting is

$$\begin{aligned}V^d(k,a)=\max_{c\geq 0,k'\in\mathcal{K}}\ & u(c)+\beta\mathbb{E}_{a'|a}(\theta V^d(k',a')+(1-\theta)V^n(0,k',a'))\\ \text{s.t. }& c+k'=(1-\kappa)ak^\alpha+(1-\delta)k\end{aligned} \tag{16}$$

and the value of repaying is

$$\begin{aligned}V^n(b,k,a)=\max_{c\geq 0,b'\in\mathcal{B},k'\in\mathcal{K}}\ & u(c)+\beta\mathbb{E}_{a'|a}V(b',k',a')\\ \text{s.t. }& c+q(b',k',a)b'+k'=ak^\alpha+(1-\delta)k+b.\end{aligned} \tag{17}$$

The most difficult part of computing this model is solving for V^n. Note the optimal policies for the problem are generally *not* monotone: An increase in bonds b or capital k may cause a substitution from b' into k' or vice-versa. This is true even if one uses a cash-at-hand formulation.

Nevertheless, binary monotonicity can be used to solve this problem by mapping it into (5) and then sorting to arrive at (6). Specifically, suppose the states (b, k) and choices (b', k') both lie in a set $\mathcal{X} = \{(b_i, k_i)\}$ having cardinality n. Creating a separate problem for each a, one may then write

$$\begin{aligned} V_a^n(i) = \max_{c \geq 0, i' \in \{1,\ldots,n\}} & \; u(c) + W_a(i') \\ \text{s.t. } & \; c = z_a(i) - w_a(i'), \end{aligned} \tag{18}$$

where $W_a(i') := \beta \mathbb{E}_{a'|a} V(b_{i'}, k_{i'}, a')$, $z_a(i) := a k_i^\alpha + (1-\delta) k_i + b_i$, and $w_a(i') := q(b_{i'}, k_{i'}, a) b_{i'} + k_{i'}$. Generally, z_a and W_a will not be increasing. However, by sorting them, one can solve the model using binary monotonicity.

E.2 Performance

As stated in the main text, binary monotonicity with sorting is $O(n \log n) + O(n' \log n')$ as either n or n' grow. While the cost depends only on the total number of points n and n', in the case of tensor grids with a fixed number of points m along each dimension, $n = m^d$ and $n' = m^{d'}$ grow quickly in m when d and d' (the dimensionality of states and choices, respectively) are bigger than 1. The cost in these terms is $O(m^{\max\{d,d'\}} \log m)$ for binary monotonicity and $O(m^{d+d'})$ for brute force. While theoretically this results in a massive improvement when $d = d' > 1$, the extreme cost of using brute force in this case means one would almost surely reformulate the problem in terms of cash-at-hand, effectively reducing d to 1.

Table 1 reports the run times and evaluation counts for brute force, simple monotonicity, and binary monotonicity for different grid sizes m.[8] In the top panel, the cash-at-hand reformulation has not been used, and so binary monotonicity vastly outperforms the other methods. For 50 points in each dimension, binary monotonicity is already 14 times faster than simple monotonicity and 39 times faster than brute force. For a 1000 points, simple monotonicity's estimated run time is two months while binary monotonicity's actual run time is only 21 minutes. A doubling of the grid sizes makes the speedup increase by roughly a factor of 4, which agrees with binary monotonicity being $O(m^2 \log m)$ and brute force being $O(m^4)$. The speedups measured in evaluation counts are even more dramatic as they exclude time spent sorting.

Reformulating the problem using cash-at-hand (with m points for the cash-at-hand state variable) makes brute force an $O(m^3)$ algorithm but has no change on binary monotonicity's asymptotics (which are still $O(m^2 \log m)$). Consequently, binary monotonicity still has an advantage, but it is much smaller. This can be seen in a comparison of the top and bottom panels of Table 1. With the cash-at-hand formulation, brute force and simple monotonicity are faster by a factor of roughly m, but binary monotonicity is only twice as fast. Overall, binary monotonicity still outperforms simple monotonicity, but by a more modest factor of 1.4 to 9.2 for run times. The better

[8] For this example, productivity follows an AR(1) with a persistence parameter of .945 and standard deviation of .025, the default cost κ is .05, the risk-free rate r is .017, the discount factor β is .952, and the capital share, risk aversion, and depreciation rate are as in the RBC calibration.

Original formulation

	Run times			Evaluations per state			
Points (m)	Brute	Simple	Binary	Brute	Simple	Binary	Speedup
50	1.73 (m)	37.1 (s)	2.64 (s)	2500	1273	14	14.1
100	26.4 (m)	9.48 (m)	9.35 (s)	10000	5100	16	60.8
250	16.1* (h)	5.83* (h)	1.10 (m)	62500*	31936*	19	317.9*
500	10.3* (d)	3.72* (d)	5.02 (m)	250000*	127919*	21	1066.8*
1000	157* (d)	57.0* (d)	21.1 (m)	1000000*	512379*	23	3890.5*

Cash-at-hand formulation

	Run times			Evaluations per state			
Points (m)	Brute	Simple	Binary	Brute	Simple	Binary	Speedup
50	3.14 (s)	1.21 (s)	0.85 (s)	2500	1304	379	1.4
100	17.5 (s)	8.03 (s)	4.16 (s)	10000	5127	860	1.9
250	4.12 (m)	1.76 (m)	31.0 (s)	62500	31723	2467	3.4
500	32.3 (m)	12.8 (m)	2.27 (m)	250000	126354	5432	5.6
1000	4.23* (h)	1.55* (h)	10.1 (m)	1000000*	503277*	11855	9.2*

Note: Run times are in seconds (s), minutes (m), hours (h), or days (d). The last column gives the run time for simple relative to binary; an * means the value is estimated; times and average evaluations are over the first 200 value function iterations.

Table 1: Run times and evaluations for the combined Arellano (2008) and RBC model

evaluation count speedups, which are in the 3.4 to 42.5 range, show that sorting costs are playing a non-trivial role. When only n or n' grow, sorting costs dominate, and here that is essentially the case because $n' = m^2$ grows much faster than $n = m$. However, the speedups measured against brute force—which may be a better benchmark since the sorting of continuation utilities is, to our knowledge, novel—are roughly twice as large.

E.3 Two-stage reformulations

Some models, such as models with adjustment costs, do not directly have the additive separability in the budget constraint of (6). However, they might when breaking the maximization problem into two stages. For instance, adding capital adjustment costs to our example results in a budget constraint

$$c + q(b', k', a)b' + k' + \xi(k' - k)^2 = ak^\alpha + (1 - \delta)k + b, \tag{19}$$

which can be written as $c = z_a(b, k, k') - w_a(b', k, k')$ where $z_a(b, k, k') := ak^\alpha + (1 - \delta)k + b$ and $w_a(b', k, k') := q(b', k', a)b' + k' + \xi(k' - k)^2$. Consequently, the problem has the form

$$\begin{aligned} V_a^n(i, j) = &\max_{c \geq 0, i', j'} u(c) + W_a(i', j') \\ &\text{s.t. } c = z_a(i, j, j') - w_a(i', j, j'), \end{aligned} \tag{20}$$

so that there is additive separability between the i and i' variables *conditional* on a j, j' pair. Binary monotonicity can be used to solve (20) by breaking it into a two-stage problem where j' is chosen in the first stage and i' in the second:

$$\begin{aligned} V_a^n(i, j) &= \max_{j'} \tilde{V}_a^n(i, j, j') \\ \tilde{V}_a^n(i, j, j') &= \max_{c \geq 0, i'} u(c) + W_a(i', j') \\ &\text{s.t. } c = z_a(i, j, j') - w_a(i', j, j'). \end{aligned} \tag{21}$$

For each j, j' combination, one can sort $z(\cdot, j, j')$ and $W(\cdot, j')$ so that the optimal policy of the second-stage problem is monotone in i. For grids of size m in each dimension, binary monotonicity with sorting can be used to solve for $\tilde{V}_a^n$ in $O(m^3 \log m)$ operations and then V_a^n in $O(m^3)$ operations. So, the total cost is $O(m^3 \log m)$, which compares favorably with brute force's $O(m^4)$.

In general, if one wants to use binary monotonicity for one choice variable (say i') but not another (say j'), a two-stage reformulation must be done. The above example illustrates one way to do this: First choose j' and make it a state variable when choosing i'. The other way is to first choose i' and make it a state variable when choosing j'. The RBC model with elastic labor supply provides an example of this latter approach, and one may consult Section C.2.3 of Appendix C for more details.

F Omitted proofs and lemmas

This appendix gives omitted proofs and lemmas. Section F.1 gives the results for the monotonicity-related sufficient conditions. Section F.2 gives proofs and lemmas showing binary monotonicity and the other algorithms work correctly. Section F.3 gives the proofs and lemmas for the cost bounds in established in Propositions 1 and 2 in the main text.

F.1 Monotonicity sufficient condition proofs and lemmas

To give the omitted sufficient condition proofs, we first give some definitions and a lemma.

F.1.1 Definitions and a lemma

Lattices are general mathematical structures. For our purposes, we need only lattices consisting of subsets of Euclidean space with the component-wise ordering, i.e., $x \leq y$ for $x, y \in \mathbb{R}^n$ if $x_j \leq y_j$ for all j where z_j denotes the jth component of z. In this context, the join operation $\vee$ gives the component-wise maximum, namely, $x \vee y = (\max\{x_1, y_1\}, \ldots, \max\{x_n, y_n\})$. Likewise, the meet operation $\wedge$ gives the component-wise minimum, $x \wedge y = (\min\{x_1, y_1\}, \ldots, \min\{x_n, y_n\})$. A lattice consists of a set $X \subset \mathbb{R}^n$ with the component-wise ordering such that $x, y \in X$ implies $x \vee y, x \wedge y \in X$. Note that if $X \subset \mathbb{R}$, it constitutes a lattice with our ordering: $x, y \in X$ implies $\min\{x, y\}, \max\{x, y\} \in X$. A function $f : X \to \mathbb{R}$ where X is a lattice is said to be supermodular (submodular) if $f(x) + f(y) \leq (\geq) f(x \wedge y) + f(x \vee y)$ for all $x, y \in X$; if the inequality is strict for all x and y that cannot be ordered, then the function is strictly supermodular (submodular).

For part of Lemma 2, we will need a slightly broader definition of increasing differences than what was given in the main text:

Definition 4. *Let $X \subset \mathbb{R}^n$. Use the notation (x_{-ij}, y_i, y_j) to denote the vector x but with the ith and jth component replaced with the ith and jth component of y, respectively. A function $f : X \to \mathbb{R}$ has* increasing differences *on X if for all i, j with $i \neq j$ and for all y_i, y_j having $x_i \leq y_i$ and $x_j \leq y_j$ (such that $(x_{-ij}, y_i, x_j), (x_{-ij}, x_i, x_j), (x_{-ij}, y_i, y_j), (x_{-ij}, x_i, y_j) \in X$) one has*

$$f(x_{-ij}, y_i, x_j) - f(x_{-ij}, x_i, x_j) \leq f(x_{-ij}, y_i, y_j) - f(x_{-ij}, x_i, y_j).$$

The function f has decreasing differences *if $f(x_{-ij}, y_i, x_j) - f(x_{-ij}, x_i, x_j) \geq f(x_{-ij}, y_i, y_j) - f(x_{-ij}, x_i, y_j)$. The differences are* strict *if the inequality holds strictly (whenever $x_i < y_i$ and $x_j < y_j$).*

Note that this is equivalent to having increasing differences—as defined in the main text—for all pairs of components. Any univariate function has increasing differences because the condition requires $i \neq j$.

We will also need to appeal to the following partial equivalence between increasing differences and supermodularity.

Lemma 3. *Suppose $X \subset \mathbb{R}^n$ is a lattice. If f is (strictly) supermodular on X, then f has (strictly) increasing differences on X. If $X = \prod_{i=1}^n X_i$ with $X_i \subset \mathbb{R}$ for all i and f has (strictly) increasing differences on X, then f is (strictly) supermodular on X.*

Proof. Let $\times$ denote the direct product (a generalization of the Cartesian product).[9] Then Theorem 2.6.1 of Topkis (1998) gives that if X_α is a lattice for each α in a set A, X is a sublattice of $\times_{\alpha \in A} X_\alpha$, and $f(x)$ is (strictly) supermodular on X, then $f(x)$ has (strictly) increasing differences on X. Taking $A = \{1, \ldots, n\}$ and $X_\alpha = \mathbb{R}$ for all $\alpha \in A$ gives $\times_{\alpha \in A} X_\alpha = \mathbb{R}^n$ (which is example 2.2.1 part c on Topkis, 1998, p. 12). Consequently, X is a sublattice of $\mathbb{R}^n$ and the theorem applies to show the (strict) supermodularity of f on X implies (strictly) increasing differences of f on X.

Corollary 2.6.1 of Topkis (1998) gives that if X_i is a chain (by definition, a partially ordered set that contains no unordered pairs of elements) for $i = 1, \ldots, n$ and f has (strictly) increasing differences on $\times_{i=1}^n X_i$, then f is (strictly) supermodular on $\times_{i=1}^n X_i$. Since $X_i \subset \mathbb{R}$, it is a chain, and the direct product $\times_{i=1}^n X_i$ is just the Cartesian product $\prod_{i=1}^n X_i$. Hence (strictly) increasing differences on X implies (strict) supermodularity on X. □

F.1.2 Proofs

Proof of Proposition 3

Proof. Because $\mathcal{I}' \subset \mathbb{R}$, it is a lattice. Additionally, $-\pi(i, i')$ has decreasing differences and is trivially submodular in i' (as well as supermodular). So Theorem 6.1 of Topkis (1978) gives that $\arg\min_{i' \in I'(i)} -\pi(i, i')$ is ascending on the set of i such that a solution exists. Theorem 6.3 strengthens this to strongly ascending when $-\pi(i, i')$ has strictly decreasing differences. Noting $G(i) := \arg\max_{i' \in I'(i)} \pi(i, i') = \arg\min_{i' \in I'(i)} -\pi(i, i')$ then gives the result. □

Proof of Lemma 1

Proof. To have I' ascending on $\mathcal{I}$ one needs $i_1 < i_2$, $i_1' \in I'(i_1)$, and $i_2' \in I'(i_2)$ to imply $\min\{i_1', i_2'\} \in I'(i_1)$ and $\max\{i_1', i_2'\} \in I'(i_2)$. Since I' is increasing, $i_1' \in I'(i_2)$ and so $\max\{i_1', i_2'\} \in I'(i_2)$. If $i_2' \geq i_1'$, then one has $\min\{i_1', i_2'\} = i_1' \in I'(i_1)$. So, take $i_2' < i_1'$.

We need to show that $i_2' \in I'(i_1)$ for $i_1 < i_2$ and $i_2' < i_1'$. Pick an arbitrary m and suppress dependence on it. Then

$$\begin{aligned} h(i_2, i_2') - h(i_1, i_2') &\leq h(i_2, i_1') - h(i_1, i_1') \\ &\leq h(i_2, i_1') \\ &\leq h(i_2, i_2') \end{aligned}$$

where the first line follows from increasing differences, the second from $i_1' \in I'(i_1)$ so that $h(i_1, i_1') \geq 0$, and the third from h being decreasing in i'. Consequently $-h(i_1, i_2') \leq 0$ which gives $h(i_1, i_2') \geq 0$.

[9] Topkis (1998) defines the direct product, which he denotes by $\times$, in this way: "If X_α is a set for each α in a set A, then the direct product of these sets X_α is the product set $\times_{\alpha \in A} X_\alpha = \{x = (x_\alpha : \alpha \in A) : x_\alpha \in X_\alpha \text{ for each } \alpha \in A\}$" (p. 12). The notation $(x_\alpha : \alpha \in A)$ gives a vector that "consists of a component x_α for each $\alpha \in A$" (p. 12). In words, $\times_{\alpha \in A} X_\alpha$ is the set of vectors that can be formed under the restriction that each α component has to lie in X_α.

Since the m was arbitrary, this holds for all m and so $i'_2 \in I'(i_1)$. Thus, $\min\{i'_1, i'_2\} \in I'(i_1)$ and so I' is ascending on $\mathcal{I}$. □

Proof of Lemma 2

Proof. For (a) and (b), we prove (b) which implies (a). Let $(i_1, i'_1), (i_2, i'_2) \in S$ with $i_1 < i_2$ and $i'_1 < i'_2$ but otherwise arbitrary. Then

$$\begin{aligned} f(i_2, i'_j) - f(i_1, i'_j) &= p(i_2) + q(i'_j) + r(i_2)s(i'_j) - p(i_1) - q(i'_j) - r(i_1)s(i'_j) \\ &= p(i_2) - p(i_1) + (r(i_2) - r(i_1))s(i'_j). \end{aligned}$$

So, $f(i_2, i'_1) - f(i_1, i'_1) \leq f(i_2, i'_2) - f(i_1, i'_2)$ if and only if

$$\begin{aligned} p(i_2) - p(i_1) + (r(i_2) - r(i_1))s(i'_1) &\leq p(i_2) - p(i_1) + (r(i_2) - r(i_1))s(i'_2) \\ \Leftrightarrow 0 &\leq (r(i_2) - r(i_1))(s(i'_2) - s(i'_1)), \end{aligned}$$

which holds because r and s are either both increasing or both decreasing.

For (c), let $(i_1, i'_1), (i_2, i'_2) \in S$ with $i_1 < i_2$ and $i'_1 < i'_2$ but otherwise arbitrary. Then f has increasing differences if and only if

$$g(i_2, i'_1) - g(i_1, i'_1) \leq g(i_2, i'_2) - g(i_1, i'_2)$$

since g agrees with f on S. Because g is C^2, this holds if

$$\int_{[i_1, i_2]} g_1(\theta, i'_1) d\theta \leq \int_{[i_1, i_2]} g_1(\theta, i'_2) d\theta \Leftrightarrow 0 \leq \int_{[i_1, i_2]} (g_1(\theta, i'_2) - g_1(\theta, i'_1)) d\theta$$

Again, by g being C^2, this holds if

$$0 \leq \int_{[i_1, i_2]} \int_{[i'_1, i'_2]} g_{12}(\theta, \theta') d\theta' d\theta$$

Because L is assumed to be a hypercube containing S and $g_{12} \geq 0$ on L, this holds.

For (d), let $(i_1, i'_1), (i_2, i'_2) \in S$ with $i_1 < i_2$ and $i'_1 < i'_2$ but otherwise arbitrary. Then f has increasing differences if and only if

$$\begin{aligned} &\sum_k \alpha_k f_k(i_2, i'_1) - \sum_k \alpha_k f_k(i_1, i'_1) \leq \sum_k \alpha_k f_k(i_2, i'_2) - \sum_k \alpha_k f_k(i_1, i'_2) \\ &\Leftrightarrow \sum_k \alpha_k (f_k(i_2, i'_1) - f_k(i_1, i'_1)) \leq \sum_k \alpha_k (f_k(i_2, i'_2) - f_k(i_1, i'_2)). \end{aligned}$$

A sufficient condition for this is that, for all k, f_k has increasing differences so that $f_k(i_2, i'_1) - f_k(i_1, i'_1) \leq f_k(i_2, i'_2) - f_k(i_1, i'_2)$.

For (e) and (f), let $(i_1, i'_1), (i_2, i'_2) \in S$ with $i_1 < i_2$ and $i'_1 < i'_2$ but otherwise arbitrary. The

composition $h \circ g$ has increasing differences on S if and only if

$$h(g(i_2, i_1')) - h(g(i_1, i_1')) \leq h(g(i_2, i_2')) - h(g(i_1, i_2')) \tag{22}$$

Because g is increasing in i, $g(i_2, i') - g(i_1, i') \geq 0$. Then because h is C^2, (22) is equivalent to

$$\int_0^{g(i_2,i_1')-g(i_1,i_1')} h'(g(i_1, i_1') + \theta)d\theta \leq \int_0^{g(i_2,i_2')-g(i_1,i_2')} h'(g(i_1, i_2') + \theta)d\theta.$$

Because g has increasing differences, $g(i_2, i_1') - g(i_1, i_1') \leq g(i_2, i_2') - g(i_1, i_2')$. Hence, this is equivalent to

$$\int_0^{g(i_2,i_1')-g(i_1,i_1')} \left(h'(g(i_1, i_1') + \theta) - h'(g(i_1, i_2') + \theta)\right) d\theta \leq \int_{g(i_2,i_1')-g(i_1,i_1')}^{g(i_2,i_2')-g(i_1,i_2')} h'(g(i_1, i_2') + \theta)d\theta.$$

Because $h' > 0$, the right-hand side is positive. So, a sufficient condition for this to hold is that the left-hand side be negative, which is true if $h'(g(i_1, i_1') + \theta) \leq h'(g(i_1, i_2') + \theta)$ for all positive θ. In (f), g is increasing in its second argument, so $g(i_1, i_1') + \theta \leq g(i_1, i_2') + \theta$, and, because $h'' > 0$, this holds. In (g), g is decreasing in its second argument, so $g(i_1, i_1') + \theta \geq g(i_1, i_2') + \theta$, and, because $h'' < 0$, this holds. So, $h \circ g$ has increasing differences.

For (g), let $(i_1, i_1'), (i_2, i_2') \in S$ with $i_1 < i_2$ and $i_1' < i_2'$ but otherwise arbitrary. Then because h is increasing in i and because g has increasing differences,

$$g(h(i_2, \varepsilon), i_1') - g(h(i_1, \varepsilon), i_1') \leq g(h(i_2, \varepsilon), i_2') - g(h(i_1, \varepsilon), i_2')$$

for any $\varepsilon \in E$. Integrating,

$$\int_E \left(g(h(i_2, \varepsilon), i_1') - g(h(i_1, \varepsilon), i_1')\right) dF(\varepsilon) \leq \int_E g(h(i_2, \varepsilon), i_2') - g(h(i_1, \varepsilon), i_2')dF(\varepsilon).$$

From $f(i, i') = \int_E g(h(i, \varepsilon), i')dF(\varepsilon)$, this says

$$f(i_2, i_1') - f(i_1, i_1') \leq f(i_2, i_2') - f(i_1, i_2'),$$

which establishes that f has increasing differences.

For (h), note that the pairwise increasing differences of g in i, i' and i, x and i', x gives, by definition, that g has increasing differences on $\mathcal{I} \times \mathcal{I}' \times X$. So, g is supermodular on the lattice $\mathcal{I} \times \mathcal{I}' \times X$ by Lemma 3. Since the graph of Γ is assumed to be a lattice, it is a sublattice of $S \times X$. Last, because g is supermodular, $-g$ is submodular. Consequently, Theorem 4.3 of Topkis (1978) applies to show that $\min_{x \in \Gamma(i,i')} -g(i, i', x)$ is submodular. Therefore, $-\min_{x \in \Gamma(i,i')} -g(i, i', x)$ is supermodular, and this equals $\max_{x \in \Gamma(i,i')} g(i, i', x)$, which by definition is $f(i, i')$. So, f is supermodular on the lattice S, which implies f has increasing differences on S by Lemma 3. □

Proof of Proposition 4

Proof. Let $a < b$ with $a, b \in \mathcal{I}$. Define the feasible set as $\Gamma(i) := \{i' \in \mathcal{I}' | c(i, i') \geq 0\}$. Let $g_1 \in G(a)$ and $g_2 \in G(b)$.

To establish that G is ascending, it is sufficient to show that $g_1 > g_2$ implies $g_1 \in G(b)$ and $g_2 \in G(a)$. To establish that G is strongly ascending, it is sufficient to show that $g_1 > g_2$ gives a contradiction. So, suppose $g_1 > g_2$. Unless explicitly stated, we only assume c is weakly increasing in i, has weakly increasing differences, and that W is weakly increasing.

First, we will show $c(a, g_2) \geq c(a, g_1) \geq 0$ and $c(b, g_2) \geq c(b, g_1) \geq 0$ for W weakly increasing and $c(a, g_2) > c(a, g_1) \geq 0$ and $c(b, g_2) > c(b, g_1) \geq 0$ for W strictly increasing. To see this, note that Γ is increasing. Consequently, $g_1 \in \Gamma(b)$ (so $c(b, g_1) \geq 0$) and hence $g_2 \in G(b)$ implies

$$u(c(b, g_2)) + W(g_2) \geq u(c(b, g_1)) + W(g_1). \tag{23}$$

Because $g_2 < g_1$ and W is weakly (strictly) increasing, this implies $c(b, g_2) \geq (>) c(b, g_1)$. So, exploiting weakly increasing differences, $0 \geq (>) c(b, g_1) - c(b, g_2) \geq c(a, g_1) - c(a, g_2)$ for W weakly (strictly) increasing. Also using $g_1 \in \Gamma(a)$, $c(a, g_2) \geq (>) c(a, g_1) \geq 0$ for W weakly (strictly) increasing. For use below, note also that because $g_1 \in G(a)$ and $g_2 \in \Gamma(a)$,

$$u(c(a, g_1)) + W(g_1) \geq u(c(a, g_2)) + W(g_2). \tag{24}$$

Combining (23) and (24),

$$u(c(b, g_2)) + W(g_2) - u(c(b, g_1)) - W(g_1) \geq 0 \geq u(c(a, g_2)) + W(g_2) - u(c(a, g_1)) - W(g_1),$$

which implies

$$u(c(b, g_2)) - u(c(b, g_1)) \geq u(c(a, g_2)) - u(c(a, g_1)). \tag{25}$$

As established above, $c(b, g_2) \geq c(b, g_1)$ and $c(a, g_2) \geq c(a, g_1)$. Using this and the differentiability of u, (25) is equivalent to

$$\int_0^{c(b,g_2)-c(b,g_1)} u'(c(b, g_1) + \theta) d\theta \geq \int_0^{c(a,g_2)-c(a,g_1)} u'(c(a, g_1) + \theta) d\theta.$$

Because of weakly increasing differences, $c(a, g_1) - c(a, g_2) \leq c(b, g_1) - c(b, g_2)$ or, equivalently, $c(b, g_2) - c(b, g_1) \leq c(a, g_2) - c(a, g_1)$. Moreover, $c(b, g_2) \geq c(b, g_1)$. So, the above inequality is equivalent to

$$0 \geq \int_{c(b,g_2)-c(b,g_1)}^{c(a,g_2)-c(a,g_1)} u'(c(a, g_1) + \theta) d\theta + \int_0^{c(b,g_2)-c(b,g_1)} (u'(c(a, g_1) + \theta) - u'(c(b, g_1) + \theta)) d\theta$$

Because c is weakly increasing in i and u is concave, the second integral is positive. The first must also be positive. Hence, the inequality holds if and only if

$$c(a, g_2) - c(a, g_1) = c(b, g_2) - c(b, g_1) \tag{C1}$$

$$\text{AND} \quad c(b, g_2) = c(b, g_1) \text{ or } c(a, g_1) = c(b, g_1) \tag{C2}$$

Now, consider the claims again. For the second claim, we seek a contradiction. A contradiction obtains if c has strictly increasing differences as then (C1) is violated. Alternatively, if W is strictly increasing and c is strictly increasing in i, (C2) will be violated because $c(b, g_2) > c(b, g_1)$ and $c(a, g_1) > c(b, g_1)$.

For the first claim, we want to show $g_1 \in G(b)$ and $g_2 \in G(a)$. (C2) implies either $c(b, g_2) = c(b, g_1)$ and/or $c(a, g_1) = c(b, g_1)$. Consider the cases separately with $c(b, g_2) = c(b, g_1)$ first. Then (C1) gives $c(a, g_2) - c(a, g_1) = c(b, g_2) - c(b, g_1)$. So, $c(a, g_2) = c(a, g_1)$. Hence, the choices give the same consumption at a and b. So, the continuation utility must be the same: (23) implies $W(g_2) \geq W(g_1)$ and (24) implies $W(g_1) \geq W(g_2)$. So, with the same consumption and choice utilities, $g_1 \in G(a)$ gives $g_2 \in G(a)$ and $g_2 \in G(b)$ gives $g_1 \in G(b)$.

Now consider the second case where $c(a, g_1) = c(b, g_1)$. Because (C1) gives $c(a, g_2) - c(a, g_1) = c(b, g_2) - c(b, g_1)$, replacing $c(a, g_1)$ with $c(b, g_1)$ gives $c(a, g_2) - c(b, g_1) = c(b, g_2) - c(b, g_1)$ or $c(a, g_2) = c(b, g_2)$. Then

$$\begin{aligned} &u(c(a, g_1)) + W(g_1) \geq u(c(a, g_2)) + W(g_2) \\ \Leftrightarrow\ &u(c(b, g_1)) + W(g_1) \geq u(c(b, g_2)) + W(g_2) \end{aligned} \tag{26}$$

where the first line follows from the optimality of $g_1 \in G(a)$ and the second from $c(a, g_1) = c(b, g_1)$ and $c(a, g_2) = c(b, g_2)$. Consequently, since $g_2 \in G(b)$ and (26) shows g_1 delivers weakly higher utility at b, $g_1 \in G(b)$.

To establish $g_2 \in G(a)$, the argument is similar. We have $c(a, g_1) = c(b, g_1)$ and $c(a, g_2) = c(b, g_2)$. Because we have shown $g_1 \in G(b)$, $u(c(b, g_1)) + W(g_1) = u(c(b, g_2)) + W(g_2)$. Replacing $c(b, g_1)$ with $c(a, g_1)$ and $c(b, g_2)$ with $c(a, g_2)$, this becomes

$$u(c(a, g_1)) + W(g_1) = u(c(a, g_2)) + W(g_2).$$

Consequently, $g_1 \in G(a)$ implies $g_2 \in G(a)$. □

Proof of Proposition 5

Proof. Because f is assumed to be differentiable (left and right) on the closure of S, Theorem 1 of Milgrom and Segal (2002) gives that $f_i(i, i') = u_1(g(x^*(i, i'), i, i'))g_2(x^*(i, i'), i, i')$ on S.

To show increasing differences, we need to establish that

$$f(i_2, i_1') - f(i_1, i_1') \leq f(i_2, i_2') - f(i_1, i_2')$$

for $(i_1, i_1'), (i_2, i_2') \in S$ with $i_1 \leq i_2$ and $i_1' \leq i_2'$. Since f is C^1 in i, this is equivalent to

$$0 \leq \int_{i_1}^{i_2} (f_i(\theta, i_2') - f_i(\theta, i_1'))d\theta.$$

Hence, if f_i is increasing in i', then increasing difference holds. Defining $x_j^* := x^*(i, i'_j)$, then f_i is increasing in i' if

$$\begin{aligned}0 &\leq u_1(g(x_2^*, i, i'_2))g_2(x_2^*, i, i'_2) - u_1(g(x_1^*, i, i'_1))g_2(x_1^*, i, i'_1) \\ &= u_1(g(x_2^*, i, i'_2))(g_2(x_2^*, i, i'_2) - g_2(x_1^*, i, i'_1)) + (u_1(g(x_2^*, i, i'_2)) - u_1(g(x_1^*, i, i'_1)))g_2(x_1^*, i, i'_1)\end{aligned}$$

Since $u_1 \geq 0$ and $g_2(x^*(i, i'), i, i')$ is increasing in i', the first term is positive. Since $u_{11} \leq 0$, $g_2(x^*(i, i'), i, i') \geq 0$, and $g(x^*(i, i'), i, i')$ is decreasing in i', the second term is positive. So, f has increasing differences. □

F.2 Algorithm correctness proofs and lemmas

We now give the omitted proofs and lemmas pertaining to the algorithm correctness.

Let the objective function $\tilde{\pi}(i, i')$, the maximum $\tilde{\Pi}(i)$, and the feasible choice set $I'(i)$ be as in (7). Assume that $I'(i) \subset \{1, \ldots, n'\}$—which may be empty—is monotonically increasing, and define $I := \{i \in \{1, \ldots, n\} | I'(i) \neq \emptyset\}$ so that $i \in I$ has a feasible solution. For all $i \in I$, define $\tilde{G}(i) := \arg\max_{i' \in I'(i)} \tilde{\pi}(i, i')$. Let π be defined via (8)—where $\underline{\pi}$ is such that $\tilde{\pi}(i, i') > \underline{\pi}$ for all $i \in I$ and $i' \in I'(i)$—with $\Pi(i) := \max_{i' \in \{1, \ldots, n'\}} \pi(i, i')$ and $G(i) := \arg\max_{i' \in \{1, \ldots, n'\}} \pi(i, i')$. Note that by construction, $i' = 1$ is optimal whenever there is no feasible choice and $i' \in I'(i)$ is always preferable to $i' \notin I'(i)$ when a feasible choice exists.

Lemma 4 establishes the mathematical equivalence of these problems (but does not say that the algorithms applied to (1) deliver a correct solution to (7)).

Lemma 4. *All of the following are true:*

1. $\Pi(i) = \tilde{\Pi}(i)$ *for all* $i \in I$.
2. $G(i) = \tilde{G}(i)$ *for all* $i \in I$.
3. *If* $\tilde{G}$ *is ascending on* I*, then* G *is ascending on* $\{1, \ldots, n\}$.

Proof. The first claim is an implication of the second claim. For the proof of the second claim, let $i \in I$. Then $I'(i) \neq \emptyset$. Infeasible choices, i.e., $i' \in \{1, \ldots, n'\} \setminus I'(i)$, are strictly suboptimal in the Π problem (1) because any feasible choice $j' \in I'(i)$ delivers $\pi(i, j') > \underline{\pi} = \pi(i, i')$. Hence,

$$G(i) = \underset{i' \in \{1, \ldots, n'\}}{\arg\max}\ \pi(i, i') = \underset{i' \in I'(i)}{\arg\max}\, \pi(i, i') = \underset{i' \in I'(i)}{\arg\max}\, \tilde{\pi}(i, i') = \tilde{G}(i)$$

(where the third equality follows from the definition of π).

To show the third claim, let $\tilde{G}$ be ascending on I. Now, let $i_1 \leq i_2$ and $g_1 \in G(i_1)$ and $g_2 \in G(i_2)$. We want to show that $\min\{g_1, g_2\} \in G(i_1)$ and $\max\{g_1, g_2\} \in G(i_2)$. Clear this is the case if $g_1 \leq g_2$, so take $g_1 > g_2$. Then since $G(i) = \{1\}$ for all $i \notin I$ and $g_1 > g_2 \geq 1$, it must be that $i_1 \in I$ (otherwise, g_1 would have to be 1). Then, because $I'(i)$ is increasing, $i_2 \in I$. Hence, $G(i_1) = \tilde{G}(i_1)$ and $G(i_2) = \tilde{G}(i_2)$. So, $\tilde{G}$ ascending gives the desired result. □

Lemmas 5 and 6 establish that, for concave problems, simple and binary concavity deliver an optimal choice provided there is one in the search space.

Lemma 5. *If the problem is concave and $\pi(i,j) \geq \pi(i,j{+}1)$ for some j, then $\pi(i,j) = \max_{i' \in \{j,\ldots,n'\}} \pi(i,i')$ (j is as least as good as anything to the right of it). If $\pi(i,k-1) < \pi(i,k)$ for some k, then $\pi(i,k) = \max_{i' \in \{1,\ldots,k\}} \pi(i,i') = \max_{i' \in \{1,\ldots,k\}} \tilde{\pi}(i,i')$ (k is as least as good as anything to the left of it).*

Proof. Note that by the problem being concave (as given in Definition 3), there is some increasing function $n'(i)$ such that $I'(i) = \{1,\ldots,n'(i)\}$ for $i \in I$.

To prove $\pi(i,j) \geq \pi(i,j+1)$ implies $\pi(i,j) = \max_{i' \in \{j,\ldots,n'\}} \pi(i,i')$, consider two cases. First, suppose $i \notin I$. Then $I'(i) = \emptyset$ and $\pi(i,i') = \mathbf{1}[i' = 1]$. Consequently, for any j, $\pi(i,j) = \max_{i' \in \{j,\ldots,n'\}} \pi(i,i')$. In other words, j is weakly better than any value to the right of it.

Second, suppose $i \in I$. If $j > n'(i)$, then $\pi(i,j) = \underline{\pi} = \max_{i' \in \{j,\ldots,n'\}} \pi(i,i')$. If $j = n'(i)$, then $\pi(i,j) > \underline{\pi} = \max_{i' \in \{j+1,\ldots,n'\}} \pi(i,i')$ implying $\pi(i,j) = \max_{i' \in \{j,\ldots,n'\}} \pi(i,i')$. If $j < n'(i)$, then

$$\begin{aligned}
\max_{i' \in \{j,\ldots,n'\}} \pi(i,i') &= \max\{\max_{i' \in \{j,\ldots,n'(i)\}} \pi(i,i'), \max_{i' \in \{n'(i)+1,\ldots,n'\}} \pi(i,i')\} \\
&= \max\{\max_{i' \in \{j,\ldots,n'(i)\}} \pi(i,i'), \underline{\pi}\} \\
&= \max_{i' \in \{j,\ldots,n'(i)\}} \pi(i,i') \\
&= \max_{i' \in \{j,\ldots,n'(i)\}} \tilde{\pi}(i,i').
\end{aligned}$$

All that remains to be shown for this case is $\pi(i,j) = \max_{i' \in \{j,\ldots,n'(i)\}} \tilde{\pi}(i,i')$. Since $\pi(i,j) = \tilde{\pi}(i,j)$ and $\pi(i,j+1) = \tilde{\pi}(i,j+1)$, the hypothesis gives $\tilde{\pi}(i,j) \geq \tilde{\pi}(i,j+1)$. Because the problem is concave and $\tilde{\pi}(i,\cdot)$ is weakly decreasing from j to $j+1$, it must be weakly decreasing from j to $n'(i)$. Hence, $\tilde{\pi}(i,j) = \max_{i' \in \{j,\ldots,n'(i)\}} \tilde{\pi}(i,i')$. So, $\pi(i,j) = \tilde{\pi}(i,j) = \max_{i' \in \{j,\ldots,n'(i)\}} \tilde{\pi}(i,i') = \max_{i' \in \{j,\ldots,n'\}} \pi(i,i')$.

Now, we prove the case for $\pi(i,k-1) < \pi(i,k)$. In this case, $k-1$ and k must both be feasible, i.e. $k \leq n'(i)$, because (1) if they were both infeasible, then $\pi(i,k-1) = \underline{\pi} = \pi(i,k)$ and (2) if only k were infeasible, then $\pi(i,k-1) > \underline{\pi} = \pi(i,k)$. Given that $k-1$ and k are feasible, $\pi(i,k-1) = \tilde{\pi}(i,k-1)$ and $\pi(i,k) = \tilde{\pi}(i,k)$. Since $\tilde{\pi}(i,\cdot)$ is strictly increasing until it switches to weakly decreasing, $\tilde{\pi}(i,1) < \ldots < \tilde{\pi}(i,k-1) < \tilde{\pi}(i,k)$. Hence $\tilde{\pi}(i,k) = \max_{i' \in \{1,\ldots,k\}} \tilde{\pi}(i,i')$. Since all of $1,\ldots,k$ are feasible, $\pi(i,k) = \max_{i' \in \{1,\ldots,k\}} \pi(i,i') = \max_{i' \in \{1,\ldots,k\}} \tilde{\pi}(i,i')$. □

Lemma 6. *Suppose it is known that $G(i) \cap \{a,\ldots,b\}$ is nonempty. Then brute force applied to*

$$\max_{i' \in \{a,\ldots,b\}} \pi(i,i')$$

delivers an optimal solution, i.e., letting $\hat{g}$ be the choice the algorithm delivers, $\hat{g} \in G(i)$. Additionally, if the problem is concave, then the simple concavity and binary concavity algorithms also deliver an optimal solution.

Proof. First, suppose $i \notin I$ so that $\pi(i,i') = \mathbf{1}[i'=1]$. Then it must be that $a=1$ since $G(i)=\{1\}$. Brute force clearly finds the optimum since it checks every value of i'. Simple concavity will compare $i'=a=1$ against $i'=a+1=2$ and find $i'=2$ is strictly worse. So, it stops and gives $\hat{g}=1$, implying $\hat{g} \in G(i)=\{1\}$. Binary concavity first checks whether $b-a+1 \leq 2$. If so, it is the same as brute force. If not, it checks whether $b-a+1 \leq 3$. If so, then $b-a+1=3$ and it does a comparison of either (1) a and $m=(b+a)/2$, in which case it correctly identifies the maximum as a or (2) m and b in which case it drops b from the search space and does a brute force comparison of a and $a+1$ (when it goes to step 2). If $b-a+1>3$, it will evaluate the midpoint $m=\lfloor (a+b)/2 \rfloor$ and $m+1$ and find $\pi(i,m)=\pi(i,m+1)=0$. It will then proceed to step 2, searching for the optimum in $\{1,\ldots,m\}$ with $a=1$ and $b=m$ in the next iteration of the recursive algorithm. This proceeds until $b-a+1 \leq 3$, where it then correctly identifies the maximum (as was just discussed). Therefore, binary concavity finds a correct solution, $\hat{g} \in G(i)$.

Now, suppose $i \in I$. Because brute force will evaluate $\pi(i,\cdot)$ at every $i' \in \{a,\ldots,b\}$, it finds $\hat{g} \in G(i)$. Now, suppose the problem is concave. The simple concavity algorithm evaluates $\pi(i,i')$ at $i' \in \{a,\ldots,b\}$ sequentially until it reaches an $x \in \{a+1,\ldots,b\}$ that $\pi(i,x-1) \geq \pi(i,x)$. If this stopping rule is not triggered, then simple concavity is identical to brute force and so finds an optimal solution. So, it suffices to consider otherwise. In this case, $x-1$ satisfies the conditions for "j" in Lemma 5 and hence $\pi(i,x-1)=\max_{i' \in \{x-1,\ldots,n'\}} \pi(i,i')$. By virtue of not having stopped until $x-1$, $\pi(i,x-1) \geq \max_{i' \in \{a,\ldots,x-1\}} \pi(i,i')$. Consequently, $\pi(i,x-1) \geq \max_{i' \in \{a,\ldots,x-1\} \cup \{x-1,\ldots,n'\}} \pi(i,i') = \max_{i' \in \{a,\ldots,n'\}} \pi(i,i')$. Since a maximum is known to be in $\{a,\ldots,b\}$,

$$\Pi(i) = \max_{i' \in \{a,\ldots,b\}} \pi(i,i') \leq \max_{i' \in \{a,\ldots,n'\}} \pi(i,i') \leq \pi(i,x-1) \leq \Pi(i).$$

So, $\pi(i,x-1)=\Pi(i)$ giving $x-1 \in G(i)$.

Now consider the binary concavity algorithm. If $b \leq a+1$ (so that the size of the search space, $b-a+1$, is 1 or 2), the algorithm is the same as brute force and so finds a maximum. If $b=a+2$ (a search space of size 3), the algorithm goes to either step 3(a) or step 3(b). In step 3(a), it stops if $\pi(i,a) > \pi(i,m)$ (where $m=(a+b)/2$) taking the maximum as a and otherwise does the same as brute force. So, suppose the stopping condition is satisfied. A maximum is a as long as $\pi(i,a)=\max_{i' \in \{a,\ldots,b\}} \pi(i,i')$, which it is since a satisfies the conditions for "j" in Lemma 5. In step 3(b), it stops if $\pi(i,b) > \pi(i,m)$ taking the maximum as b and otherwise does the same as brute force. So, suppose the stopping condition is satisfied. A maximum is b as long as $\pi(i,b)=\max_{i' \in \{a,\ldots,b\}} \pi(i,i')$, which is true since b satisfies all the conditions for "k" in Lemma 5.

If $b \geq a+3$ (a search space of 4 or more), binary concavity goes to step 4 of the algorithm. In this case, it evaluates at two points $m=\lfloor (a+b)/2 \rfloor$ and $m+1$. If $\pi(i,m) \geq \pi(i,m+1)$, it assumes a maximum is in $\{a,\ldots,m\}$. Since m satisfies the conditions for "j" in Lemma 5, $\pi(i,m) \geq \max_{i' \in \{m,\ldots,b\}} \pi(i,i')$, which justifies this assumption. If $\pi(i,m) < \pi(i,m+1)$, it instead assumes a maximum is in $\{m+1,\ldots,b\}$. This again is justified since $m+1$ satisfies all the conditions for "k" in Lemma 5 and so $m+1$ is better than any value of $i' < m+1$. The algorithm repeatedly

divides $\{a, \ldots, b\}$ into either $\{a, \ldots, m\}$ or $\{m+1, \ldots, b\}$ until the size of the search space is either two or three. Since we have already shown the algorithm correctly identifies a maximum when the search space is of size two or three (i.e., $b = a + 1$ or $b = a + 2$), the algorithm correctly finds the maximum for larger search spaces as long as this subdivision stops in a finite number of iterations (since then induction can be applied). Lemma 7 shows the required number of function evaluations is finite, and so this holds. □

We now give the proof of Proposition 6, which establishes that the monotonicity and concavity algorithms deliver an optimal policy.

Proof of Proposition 6

Proof. We will show, letting $\hat{g}$ be the policy function the algorithm finds, that $\hat{g}(i) \in G(i)$ for all i, which implies $\hat{g}(i) \in \tilde{G}(i)$ for all $i \in I$ by Lemma 4. Each of the brute force, simple, and binary monotonicity algorithms can be thought of as iterating through states i (in some order that, in the case of binary monotonicity, depends on π) with a search space $\{a, \ldots, b\}$. If every state is visited and optimal choice is found at each state, then an optimal solution is found. So, it suffices to show that each of the brute force, simple, and binary monotonicity algorithms explore every state $i \in \{1, \ldots, n\}$ and at each state, the following conditions are met so that Lemma 6 can be applied: (1) $\{a, \ldots, b\} \subset \{1, \ldots, n'\}$; (2) $a \leq b$; and (3) $G(i) \cap \{a, \ldots, b\} \neq \emptyset$. An application of Lemma 6 then gives $\hat{g}(i) \in G(i)$ (provided an appropriate concavity algorithm is used).

Brute force monotonicity trivially explores all states $i \in \{1, \ldots, n\}$ sequentially. At each i, $a = 1$ and $b = n'$. Consequently, $G(i) \cap \{a, \ldots .b\} \neq \emptyset$ and Lemma 6 can be applied.

Now, we prove simple monotonicity and binary monotonicity deliver a correct solution when $\tilde{G}$ is ascending, which, by Lemma 4, gives that G is ascending.

The simple monotonicity algorithm explores all states $i \in \{1, \ldots, n\}$ sequentially always with $b = n'$ (and so $a \leq b$). For $i = 1$, $a = 1$ and so $G(1) \cap \{a, \ldots, b\} \neq \emptyset$. Consequently, Lemma 6 gives that $\hat{g}(1) \in G(1)$. Now, consider some $i > 1$ and suppose for induction that $\hat{g}(i-1) \in G(i-1)$. Because G is ascending and $\hat{g}(i-1) \in G(i-1)$, any $\dot{g} \in G(i)$ implies $\max\{\hat{g}(i-1), \dot{g}\} \in G(i)$. So, $G(i) \cap \{g(i-1), \ldots, n'\} \neq \emptyset$. Hence, Lemma 6 applies, and $\hat{g}(i) \in G(i)$ completing the induction argument.

Now consider the binary monotonicity algorithm. If $n = 1$ or $n = 2$, the algorithm is the same as simple monotonicity and so delivers a correct solution. If $n > 2$, then the algorithm first correctly identifies $\hat{g}(1)$ (by brute force) and $\hat{g}(n)$ (using the same argument as simple monotonicity). It then defines $\underline{i} = 1$ and $\overline{i} = n$ and maintains the assumption that $\hat{g}(\underline{i}) \in G(\underline{i})$ and $\hat{g}(\overline{i}) \in G(\overline{i})$.

The goal of step 2 is to find the optimal solution for all $i \in \{\underline{i}, \ldots, \overline{i}\}$. The algorithm stops at step 2(a) if $\overline{i} \leq \underline{i} + 1$, in which case this objective is clearly met since $\{\underline{i}, \ldots, \overline{i}\} = \{\underline{i}, \overline{i}\}$. If the algorithm does not stop, then it computes $\hat{g}(m)$ for $m = \lfloor (a+b)/2 \rfloor$ using the search space $\{\hat{g}(\underline{i}), \ldots, \hat{g}(\overline{i})\}$. By Lemma 6, an optimum is found as long as $G(m) \cap \{\hat{g}(\underline{i}), \ldots, \hat{g}(\overline{i})\} \neq \emptyset$. If $\hat{g}(\underline{i}) \in G(\underline{i})$ and $\hat{g}(\overline{i}) \in G(\overline{i})$, then for any $\dot{g} \in G(m)$, G ascending gives $\min\{\hat{g}(\overline{i}), \max\{\hat{g}(\underline{i}), \dot{g}\}\} \in G(m)$. So, $G(m) \cap \{\hat{g}(\underline{i}), \ldots, \hat{g}(\overline{i})\} \neq \emptyset$ if $\hat{g}(\underline{i}) \in G(\underline{i})$ and $\hat{g}(\overline{i}) \in G(\overline{i})$. This holds because of the algorithm's

maintained assumptions.[10] So, if every $i \in \{2, \ldots, n-1\}$ is the midpoint of some $(\underline{i}, \bar{i})$ after iterating some number of times, the proof is complete. In other words, since the algorithm only solves for the optimal policy at midpoints once it reaches step 2, we need to prove every state (except for $i = 1$ and $i = n$) is eventually a midpoint.

To show that every $i \in \{2, \ldots, n-1\}$ is a midpoint of some interval reached in the recursion, fix an arbitrary such i and suppose not. Define $(\underline{i}_1, \bar{i}_1) = (1, n)$. When step 2 is first reached, $i \in \{\underline{i}_1 + 1, \ldots, \bar{i}_1 - 1\}$. Now, uniquely and recursively define $(\underline{i}_k, \bar{i}_k)$ to be the one of $(\underline{i}_{k-1}, m)$ and $(m, \bar{i}_{k-1})$ with $m = \lfloor (i_{k-1} + \bar{i}_{k-1})/2 \rfloor$ such that $i \in \{\underline{i}_k + 1, \ldots, \bar{i}_k - 1\}$ (because i is assumed to never be a midpoint, this is well-defined).

Now, consider the cardinality of $\{\underline{i}_k, \ldots, \bar{i}_k\}$ defining it as $N_k = \bar{i}_k - \underline{i}_k + 1$. By construction, $i \in \{\underline{i}_k + 1, \ldots, \bar{i}_k - 1\}$ for each k. So, a contradiction is reached if $\{\underline{i}_k + 1, \ldots, \bar{i}_k - 1\} = \emptyset$ which is equivalent to $N_k \leq 2$. So, it must be that $N_k \geq 3$ for all k. If N_{k-1} is odd, then $N_k = (N_{k-1} + 1)/2$. If N_{k-1} is even, $N_k \leq N_{k-1}/2 + 1$. So, in either case $N_k \leq N_{k-1}/2 + 1$. Defining M_k recursively by $M_1 = N_1$ and $M_k = M_{k-1}/2 + 1$, one can show by induction that $N_k \leq M_k$ for all k. Because $N_k \geq 3$ for all k, $M_k \geq 3$ for all k. Hence $M_k - M_{k-1} = 1 - M_{k-1}/2 \leq 1 - 3/2 = -1/2$. Hence $M_k \leq M_{k-1} - 1/2$. Therefore, M_k will be less than three in a finite number of iterations, which gives a contradiction. □

F.3 Cost bound proofs and lemmas

We now give the cost bound proofs and supporting lemmas. Section F.3.1 establishes the performance of Heer and Maußner (2005)'s binary concavity. Section F.3.2 proves the performance of the one-state binary monotonicity as stated in Proposition 1. Section F.3.3 the performance of the two-state binary monotonicity as stated in Proposition 2.

F.3.1 Binary concavity

Lemma 7. *Consider the problem* $\max_{i' \in \{a, \ldots, a+n-1\}} \pi(i, i')$ *for any* a *and any* i*. For any* $n \in \mathbb{Z}^{++}$*, binary concavity requires no more* $2\lceil \log_2(n) \rceil - 1$ *evaluations if* $n \geq 3$ *and no more than* n *evaluations if* $n \leq 2$*.*

Proof. For $n = 1$, the algorithm computes $\pi(i, a)$ and stops, so one evaluation is required. For $n = 2$, two evaluations are required ($\pi(i, a)$ and $\pi(i, a+1)$). For $n = 3$, step 3 requires $\pi(i, m)$ to be computed and may require $\pi(i, a)$ to be computed. Then step 3(a) or step 3(b) either stop with no additional function evaluations or go to step 2 with $\max\{\mathbf{1}_a, \mathbf{1}_b\} = 1$ where, in that case, at most one additional function evaluation is required. Consequently, $n = 3$ requires at most three function evaluations, which agrees with $2\lceil \log_2(3) \rceil - 1 = 3$. So, the statement of lemma holds for $1 \leq n \leq 3$.

Now consider each $n \in \{4, 5, 6, 7\}$ for any $\mathbf{1}_a, \mathbf{1}_b$ flags. Since $n \geq 4$ the algorithm is in (or goes to) step 4. Consequently, two evaluations are required. Since the new interval is either $\{a, \ldots, m\}$ or

[10] Formally, it can be shown through induction. It is true at the first instance of step 2. Since in step 2(c) the algorithm then divides into $\{\underline{i}, \ldots, m\}$ and $\{m, \ldots, \bar{i}\}$, it is also true at the next iteration. Consequently, induction gives that a maximum is found for every midpoint.

$\{m+1,\ldots,b\}$ and $\pi(i,m)$ and $\pi(i,m+1)$ are computed in step 4, the *next* step has $\max\{\mathbf{1}_a,\mathbf{1}_b\}=1$. Now, if $n=4$, the next step must be step 2, which requires at most one additional evaluation (since $\max\{\mathbf{1}_a,\mathbf{1}_b\}=1$). Hence, the total evaluations are less than or equal to 3 (two for step 4 and one for step 2). If $n=5$, then the next step is either step 2, requiring one evaluation, or step 3, requiring two evaluations. So, the total evaluations are not more than four. If $n=6$, the next step is step 3, and so four evaluations are required. Lastly, for $n=7$, the next step is either step 3, requiring two evaluations, or step 4 (with $n=4$), requiring at most three evaluations. So, the evaluations are weakly less than $5=2+\max\{2,3\}$. Hence, for every $n=4,5,6$, and 7, the required evaluations are less than $3,4,4$, and 5, respectively. One can then verify that the evaluations are less than $2\lceil\log_2(n)\rceil-1$ for these values of n.

Now, suppose $n\geq 4$. We shall prove that the required number of evaluations is less than $\sigma(n):=2\lceil\log_2(n)\rceil-1$ by induction. We have already verified the hypothesis holds for $n\in\{4,5,6,7\}$, so consider some $n\geq 8$ and suppose the hypothesis holds for all $m\in\{4,\ldots,n-1\}$. Let i be such that $n\in[2^i+1,2^{i+1}]$. Then note that two things are true, $\lceil\log_2(n)\rceil=i+1$ and $\lceil\log_2(\lfloor\frac{n+1}{2}\rfloor)\rceil=i$.[11] Since $n\geq 4$, the algorithm is in (or proceeds to) step 4, which requires two evaluations, and then proceeds with a new interval to step 4 (again). If n is even, the new interval has size $n/2$. If n is odd, the new interval either has a size of $(n+1)/2$ or $(n-1)/2$. So, if n is even, no more than $2+\sigma(n/2)$ evaluations are required; if n is odd, no more than $2+\max\{\sigma((n+1)/2),\sigma((n-1)/2)\}=2+\sigma((n+1)/2)$ evaluations are required. The even and odd case can then be handled simultaneously with the bound $2+\sigma(\lfloor\frac{n+1}{2}\rfloor)$. Manipulating this expression using the previous observation that $\lceil\log_2(n)\rceil=i+1$ and $\lceil\log_2(\lfloor\frac{n+1}{2}\rfloor)\rceil=i$,

$$\begin{aligned}2+\sigma(\lfloor\frac{n+1}{2}\rfloor)&=2+2\lceil\log_2\lfloor\frac{n+1}{2}\rfloor\rceil-1\\&=2+2i-1\\&=2\lceil\log_2(n)\rceil-1.\end{aligned}$$

Hence, the proof by induction is complete. □

F.3.2 Binary monotonicity in one dimension

In proving the performance on the one-state binary monotonicity algorithm, we allow for many maximization techniques by characterizing the algorithm's properties conditional on a monotonically increasing $\sigma:\mathbb{Z}^{++}\to\mathbb{Z}^{+}$ that bounds the evaluation count required to solve (3).

Because of the recursive nature of binary monotonicity, the π evaluation bound for general σ is also naturally recursive. In Proposition 7, we will show the algorithm's cost is $2\sigma(n')+M_\sigma(n,n')$ where the function M_σ is defined as follows:

Definition 5. *For any $\sigma:\mathbb{Z}^{++}\to\mathbb{Z}^{+}$, define $M_\sigma:\{2,3,\ldots\}\times\mathbb{Z}^{+}\to\mathbb{Z}^{+}$ recursively by $M_\sigma(z,\gamma)=$*

[11] The proof is as follows. Both $\lceil\log_2(\cdot)\rceil$ and $\lceil\log_2(\lfloor\cdot\rfloor)\rceil$ are weakly increasing functions. So $n\in[2^i+1,2^{i+1}]$ implies $\lceil\log_2(n)\rceil\in[\lceil\log_2(2^i+1)\rceil,\lceil\log_2(2^{i+1})\rceil]=[i+1,i+1]$. Likewise, $\lceil\log_2(\lfloor\frac{n+1}{2}\rfloor)\rceil\in[\lceil\log_2(\lfloor\frac{2^i+1+1}{2}\rfloor)\rceil,\lceil\log_2(\lfloor\frac{2^{i+1}+1}{2}\rfloor)\rceil]=[i,i]$.

0 *if* $z = 2$ *or* $\gamma = 0$ *and*

$$M_\sigma(z,\gamma) = \sigma(\gamma) + \max_{\gamma' \in \{1,\dots,\gamma\}} \left\{ M_\sigma(\lfloor \frac{z}{2} \rfloor + 1, \gamma') + M_\sigma(\lfloor \frac{z}{2} \rfloor + 1, \gamma - \gamma' + 1) \right\}$$

for $z > 2$ *and* $\gamma > 0$.

Now, we establish that M_σ is increasing.

Lemma 8. *For any* σ, $M_\sigma(z,\gamma)$ *is weakly increasing in* z *and* γ.

Proof. Fix a σ and suppress dependence on it. First, we will show $M(z,\gamma)$ is weakly increasing in γ for every z. The proof is by induction. For $z = 2$, $M(2,\cdot) = 0$. For $z = 3$, $M(3,\gamma) = \sigma(\gamma)$ which is weakly increasing in γ. Now consider some $z > 3$ and suppose $M(y,\cdot)$ is weakly increasing for all $y \leq z - 1$. For $\gamma_2 > \gamma_1$,

$$\begin{aligned} M(z,\gamma_1) &= \sigma(\gamma_1) + \max_{\gamma' \in \{1,\dots,\gamma_1\}} \left\{ M(\lfloor \frac{z}{2} \rfloor + 1, \gamma') + M(\lfloor \frac{z}{2} \rfloor + 1, \gamma_1 - \gamma' + 1) \right\} \\ &\leq \sigma(\gamma_2) + \max_{\gamma' \in \{1,\dots,\gamma_2\}} \left\{ M(\lfloor \frac{z}{2} \rfloor + 1, \gamma') + M(\lfloor \frac{z}{2} \rfloor + 1, \gamma_2 - \gamma' + 1) \right\} \\ &= M(z,\gamma_2) \end{aligned}$$

where the inequality is justified by σ being increasing and the induction hypothesis giving $M(\lfloor \frac{z}{2} \rfloor + 1, \cdot)$ as an increasing function (note $\lfloor \frac{z}{2} \rfloor + 1 \leq z - 1$ for all $z > 3$).

Now we will show $M(z,\gamma)$ is increasing in z for every γ. The proof is by induction. First, note that $M(2,\gamma) = 0 \leq \sigma(\gamma) = M(3,\gamma)$ for all $\gamma > 0$ and $M(2,\gamma) = 0 = M(3,\gamma)$ for $\gamma = 0$. Now, consider some $k > 3$ and suppose that for any $z_1, z_2 \leq k-1$ with $z_1 \leq z_2$ that $M(z_1,\gamma) \leq M(z_2,\gamma)$ for all γ. The goal is to show that for any $z_1, z_2 \leq k$ with $z_1 \leq z_2$ that $M(z_1,\gamma) \leq M(z_2,\gamma)$ for all γ. So, consider such $z_1, z_2 \leq k$ with $z_1 \leq z_2$. If $\gamma = 0$, then $M(z_1,\gamma) = 0 = M(z_2,\gamma)$, so take $\gamma > 0$. Then

$$\begin{aligned} M(z_1,\gamma) &= \sigma(\gamma) + \max_{\gamma' \in \{1,\dots,\gamma\}} \left\{ M(\lfloor \frac{z_1}{2} \rfloor + 1, \gamma') + M(\lfloor \frac{z_1}{2} \rfloor + 1, \gamma - \gamma' + 1) \right\} \\ &\leq \sigma(\gamma) + \max_{\gamma' \in \{1,\dots,\gamma\}} \left\{ M(\lfloor \frac{z_2}{2} \rfloor + 1, \gamma') + M(\lfloor \frac{z_2}{2} \rfloor + 1, \gamma - \gamma' + 1) \right\} \\ &= M(z_2,\gamma). \end{aligned}$$

The inequality obtains since $\lfloor \frac{z_i}{2} \rfloor + 1 \leq k - 1$ for all i (which is true since even if $z_i = k$, one has $\lfloor k/2 \rfloor + 1 \leq k - 1$ by virtue of $k > 3$). So, the induction hypothesis gives $M(\lfloor \frac{z_1}{2} \rfloor + 1, \cdot) \leq M(\lfloor \frac{z_2}{2} \rfloor + 1, \cdot)$, and the proof by induction is complete. □

Now we can give a cost bound for the one-state binary monotonicity algorithm:

Proposition 7. *Let* $\sigma : \mathbb{Z}^{++} \to \mathbb{Z}^{+}$ *be an upper bound on the number of* π *evaluations required to solve (3). Then, the algorithm requires at most* $2\sigma(n') + M_\sigma(n,n')$ *evaluations for* $n \geq 2$ *and* $n' \geq 1$.

Proof. Since g is the policy function associated with (1), $g : \{1, \ldots, n\} \to \{1, \ldots, n'\}$. By monotonicity, g is weakly increasing. Define $N : \{1, \ldots, n\}^2 \to \mathbb{Z}^+$ by

$$N(a, b) = M(b - a + 1, g(b) - g(a) + 1)$$

noting that this is well-defined (based on the definition of M) whenever $b > a$. Additionally, define a sequence of sets $\mathcal{I}_k$ for $k = 1, \ldots, n - 1$ by

$$\mathcal{I}_k := \{(\underline{i}, \bar{i}) | \bar{i} = \underline{i} + k \text{ and } \underline{i}, \bar{i} \in \{1, \ldots, n\}\}.$$

Note that for any $k \in \{1, \ldots, n-1\}$, $\mathcal{I}_k$ is nonempty and $N(a, b)$ is well-defined for any $(a, b) \in \mathcal{I}_k$.

We shall now prove that for any $k \in \{1, \ldots, n-1\}$, $(\underline{i}, \bar{i}) \in \mathcal{I}_k$ implies $N(\underline{i}, \bar{i})$ is an upper bound on the number of evaluations of π required by the algorithm in order to compute the optimal policy for all $i \in \{\underline{i}, \ldots, \bar{i}\}$ when $g(\underline{i})$ and $g(\bar{i})$ are known. If true, then beginning at step 2 in the algorithm (which assumes $g(\underline{i})$ and $g(\bar{i})$ are known) with $(\underline{i}, \bar{i}) \in \mathcal{I}_k$, $N(\underline{i}, \bar{i})$ is an upper bound on the number of π evaluations.

The argument is by induction. First, consider $k = 1$. For any $(a, b) \in \mathcal{I}_1$, the algorithm terminates at step 2(a). Consequently, the number of required π evaluations is zero, which is the same as $N(a, b) = M(b - a + 1, g(b) - g(a) + 1) = M(2, g(b) - g(a) + 1) = 0$ (recall $M(2, \cdot) = 0$).

Now, consider some $k \in \{2, \ldots, n-1\}$ and suppose the induction hypothesis holds for all j in $1, \ldots, k-1$. That is, assume for all j in $1, \ldots, k-1$ that $(\underline{i}, \bar{i}) \in \mathcal{I}_j$ implies $N(\underline{i}, \bar{i})$ is an upper bound on the number of required π evaluations when $g(\underline{i})$ and $g(\bar{i})$ are known. We shall show it holds for k as well.

Consider any $(\underline{i}, \bar{i}) \in \mathcal{I}_k$ with $g(\underline{i})$ and $g(\bar{i})$ are known. Since $\bar{i} > \underline{i} + 1$, the algorithm does not terminate at step 2(a). In step 2(b), to compute $g(m)$ (where $m := \lfloor \frac{\underline{i}+\bar{i}}{2} \rfloor$), one must find the maximum within the range $g(\underline{i}), \ldots, g(\bar{i})$, which requires at most $\sigma(g(\bar{i}) - g(\underline{i}) + 1)$ evaluations of π. In step 2(c), the space is then divided into $\{\underline{i}, \ldots, m\}$ and $\{m, \ldots, \bar{i}\}$.

If k is even, then m equals $\frac{\underline{i}+\bar{i}}{2}$. Since $(\underline{i}, m) \in \mathcal{I}_{k/2}$ and $g(\underline{i})$ and $g(m)$ are known, the induction hypothesis gives $N(\underline{i}, m)$ as an upper bound on the number of π evaluations needed to compute $g(\underline{i}), \ldots, g(m)$. Similarly, since $(m, \bar{i}) \in \mathcal{I}_{k/2}$ and $g(m)$ and $g(\bar{i})$ are known, $N(m, \bar{i})$ provides an upper bound on the number of π evaluations needed to compute $g(m), \ldots, g(\bar{i})$. Therefore, to compute $g(\underline{i}), \ldots, g(\bar{i})$, at most $\sigma(g(\bar{i}) - g(\underline{i}) + 1) + N(\underline{i}, m) + N(m, \bar{i})$ evaluations are required. Defining $\gamma = g(\bar{i}) - g(\underline{i}) + 1$ and $\gamma' = g(m) - g(\underline{i}) + 1$ and using the definition of m and N, we have

that the number of required evaluations is less than

$$\begin{aligned}
&\sigma(\gamma) + M(m - \underline{i} + 1, g(m) - g(\underline{i}) + 1) + M(\bar{i} - m + 1, g(\bar{i}) - g(m) + 1) \\
&= \sigma(\gamma) + M(\frac{\bar{i} + \underline{i}}{2} - \underline{i} + 1, \gamma') + M(\bar{i} - \frac{\bar{i} + \underline{i}}{2} + 1, g(\bar{i}) - g(m) + 1) \\
&= \sigma(\gamma) + M(\frac{\bar{i} - \underline{i}}{2} + 1, \gamma') + M(\frac{\bar{i} - \underline{i}}{2} + 1, g(\bar{i}) - g(m) + \gamma' - \gamma' + 1) \\
&= \sigma(\gamma) + M(\frac{\bar{i} - \underline{i}}{2} + 1, \gamma') + M(\frac{\bar{i} - \underline{i}}{2} + 1, g(\bar{i}) - g(m) + g(m) - g(\underline{i}) + 1 - \gamma' + 1) \\
&= \sigma(\gamma) + M(\frac{\bar{i} - \underline{i}}{2} + 1, \gamma') + M(\frac{\bar{i} - \underline{i}}{2} + 1, \gamma - \gamma' + 1).
\end{aligned}$$

By virtue of monotonicity, $g(m) \in \{g(\underline{i}), \ldots, g(\bar{i})\}$ and so $g(m) - g(\underline{i}) + 1 \in \{1, \ldots, g(\bar{i}) - g(\underline{i}) + 1\}$ or equivalently $\gamma' \in \{1, \ldots, \gamma\}$. Consequently, the number of function evaluations is not greater than

$$\sigma(\gamma) + \max_{\gamma' \in \{1,\ldots,\gamma\}} \left\{ M\left(\frac{\bar{i} - \underline{i}}{2} + 1, \gamma'\right) + M\left(\frac{\bar{i} - \underline{i}}{2} + 1, \gamma - \gamma' + 1\right) \right\}.$$

The case of k odd is very similar, but the divide-and-conquer algorithm splits the space unequally. If k is odd, then m equals $\frac{\underline{i} + \bar{i} - 1}{2}$. In this case $(\underline{i}, m) \in \mathcal{I}_{(k-1)/2}$ and $(m, \bar{i}) \in \mathcal{I}_{(k-1)/2+1}$.[12] Consequently, computing the policy for $\underline{i}, \ldots, \bar{i}$ takes no more than $\sigma(g(\bar{i}) - g(\underline{i}) - 1) + N(\underline{i}, m) + N(m, \bar{i})$ maximization steps. Defining γ and γ' the same as before and using the definition of m and N, we have that the required maximization steps is less than

$$\begin{aligned}
&\sigma(\gamma) + M(m - \underline{i} + 1, \gamma') + M(\bar{i} - m + 1, \gamma - \gamma' + 1) \\
&= \sigma(\gamma) + M(\frac{\underline{i} + \bar{i} - 1}{2} - \underline{i} + 1, \gamma') + M(\bar{i} - \frac{\underline{i} + \bar{i} - 1}{2} + 1, \gamma - \gamma' + 1) \\
&= \sigma(\gamma) + M\left(\frac{\bar{i} - \underline{i} + 1}{2}, \gamma'\right) + M\left(\frac{\bar{i} - \underline{i} + 1}{2} + 1, \gamma - \gamma' + 1\right).
\end{aligned}$$

Because M is increasing in the first argument, this is less than

$$\sigma(\gamma) + \max_{\gamma' \in \{1,\ldots,\gamma\}} \left\{ M\left(\frac{\bar{i} - \underline{i} + 1}{2} + 1, \gamma'\right) + M\left(\frac{\bar{i} - \underline{i} + 1}{2} + 1, \gamma - \gamma' + 1\right) \right\}.$$

Combining the bounds for k even and odd, the required number of π evaluations is less than

$$\sigma(\gamma) + \max_{\gamma' \in \{1,\ldots,\gamma\}} \left\{ M\left(\left\lfloor \frac{\bar{i} - \underline{i} + 1}{2} \right\rfloor + 1, \gamma'\right) + M\left(\left\lfloor \frac{\bar{i} - \underline{i} + 1}{2} \right\rfloor + 1, \gamma - \gamma' + 1\right) \right\} \tag{27}$$

because if k is even, then $\lfloor \frac{\bar{i} - \underline{i} + 1}{2} \rfloor = \frac{\bar{i} - \underline{i}}{2}$. Consequently, (27) gives an upper bound for any $(\underline{i}, \bar{i}) \in \mathcal{I}_k$ for $k \geq 1$ when $g(\underline{i})$ and $g(\bar{i})$ are known. If $N(\underline{i}, \bar{i})$ is less than this, then the proof by induction is

[12] To see this, note that $(\underline{i}, \bar{i}) \in \mathcal{I}_k$ implies $k = \bar{i} - \underline{i}$. To have, $(\underline{i}, m) \in \mathcal{I}_{(k-1)/2}$, it must be that $m = \underline{i} + \frac{k-1}{2}$. This holds: $\underline{i} + \frac{k-1}{2} = \underline{i} + \frac{\bar{i} - \underline{i} - 1}{2} = \underline{i} + \frac{\bar{i} + \underline{i} - 1 - \underline{i} - \underline{i}}{2} = \underline{i} + m + \frac{-2\underline{i}}{2} = m$. Similarly, to have $(m, \bar{i}) \in \mathcal{I}_{(k-1)/2+1}$, one must have $\bar{i} = m + (k-1)/2 + 1$. This also obtains: $m + \frac{(k-1)}{2} + 1 = \frac{\underline{i} + \bar{i} - 1}{2} + \frac{\bar{i} - \underline{i} - 1}{2} + 1 = \frac{2\bar{i} - 2}{2} + 1 = \bar{i}$.

complete.

Since $N(\underline{i}, \bar{i})$ is defined as $M(\bar{i} - \underline{i} + 1, g(\bar{i}) - g(\underline{i}) + 1)$, using the definitions of N and M shows

$$\begin{aligned} N(\underline{i}, \bar{i}) &= M(\bar{i} - \underline{i} + 1, g(\bar{i}) - g(\underline{i}) + 1) \\ &= M(\bar{i} - \underline{i} + 1, \gamma) \\ &= \sigma(\gamma) + \max_{\gamma' \in \{1,\ldots,\gamma\}} \left\{ M(\lfloor \frac{\bar{i} - \underline{i} + 1}{2} \rfloor + 1, \gamma') + M(\lfloor \frac{\bar{i} - \underline{i} + 1}{2} \rfloor + 1, \gamma - \gamma' + 1) \right\}. \end{aligned}$$

Consequently, $N(\underline{i}, \bar{i})$ exactly equals the value in (27), and the proof by induction is complete.

Step 1 of the algorithm requires at most $2\sigma(n')$ evaluations to compute $g(1)$ and $g(n)$. If $n = 2$, step 2 is never reached. Since $M(n, n') = 0$ in this case, $2\sigma(n') + M(n, n')$ provides an upper bound. If $n > 2$, then since $(1, n) \in \mathcal{I}_{n-1}$ and $g(1)$ and $g(n)$ known, only $N(1, n)$ additional evaluations are required. Therefore, to compute for each $i \in \{1, \ldots, n\}$, no more than $2\sigma(n') + N(1, n) = 2\sigma(n') + M(n, g(n) - g(1) + 1)$ function evaluations are needed. Lemma (8) then gives that this is less than $2\sigma(n') + M(n, n')$ since $g(n) - g(1) + 1 \leq n' - 1 + 1 = n'$. □

The preceding proposition gives a fairly tight bound. However, it is unwieldy because of the discrete nature of the problem. By bounding σ whose domain is $\mathbb{Z}^{++}$ with a $\bar{\sigma}$ whose domain is $[1, \infty)$, a more convenient bound can be found. We give this bound in Lemma 11. For its proof, we need the following two lemmas:

Lemma 9. *Define a sequence $\{m_i\}_{i=1}^{\infty}$ by $m_1 = 2$ and $m_i = 2m_{i-1} - 1$ for $i \geq 2$. Then $m_i = 2^{i-1} + 1$ and $\log_2(m_i - 1) = i - 1$ for all $i \geq 1$.*

Proof. The proof of $m_i = 2^{i-1} + 1$ for all $i \geq 1$ is by induction. For $i = 1$, m_1 is defined as 2, which equals $2^{1-1} + 1$. For $i > 1$, suppose it holds for $i - 1$. Then $m_i = 2m_{i-1} - 1 = 2[2^{i-2} + 1] - 1 = 2^{i-1} + 1$. □

Lemma 10. *Consider any $z \geq 2$. Then there exists a unique sequence $\{n_i\}_{i=1}^{I}$ such that $n_1 = 2$, $n_I = z$, $\lfloor \frac{n_i}{2} \rfloor + 1 = n_{i-1}$, and $n_i > 2$ for all $i > 1$. Moreover, $I = \lceil \log_2(z - 1) \rceil + 1$.*

Proof. The proof that a unique sequence exists is by construction. Let $z \geq 2$ be fixed. Define an infinite sequence $\{z_i\}_{i=1}^{\infty}$ recursively as follows: Define $z_i = T_i(z)$ for all $i \geq 1$ with $T_1(z) := z$ and $T_{i+1}(z) = \lfloor \frac{T_i(z)}{2} \rfloor + 1$. We now establish all of the following: $T_i(z) \geq 2$, $T_i(z) \geq T_{i+1}(z)$, and $T_i(z) > T_{i+1}(z)$ whenever $T_i(z) > 2$. As an immediate consequence, for any $z \geq 2$, there exists a unique $I(z) \geq 1$ such that $T_{I(z)} = 2$ and, for all $i < I(z)$, $T_i(z) > 2$. We also show for later use that $T_i(z)$ is weakly increasing in z for every i.

To show $T_i(z) \geq 2$, the proof is by induction. We have $T_1(z) = z$ and $z \geq 2$. Now, consider some $i > 1$ and suppose it holds for $i - 1$. Then $T_i(z) = \lfloor \frac{T_{i-1}(z)}{2} \rfloor + 1 \geq \lfloor \frac{2}{2} \rfloor + 1 = 2$.

To show $T_i(z) > T_{i+1}(z)$ whenever $T_i(z) > 2$, consider two cases. First, consider $T_i(z)$ even. Then $T_{i+1}(z) = \lfloor \frac{T_i(z)}{2} \rfloor + 1 = \frac{T_i(z)}{2} + 1$ and so $T_{i+1}(z) < T_i(z)$ as long as $T_i(z) > 2$. Second, consider $T_i(z)$ odd. Then $T_{i+1}(z) = \lfloor \frac{T_i(z)}{2} \rfloor + 1 = \frac{T_i(z)-1}{2} + 1$ and so $T_{i+1}(z) < T_i(z)$ as long as $T_i(z) > 1$.

To show that $T_i(z) \geq T_{i+1}(z)$, all we need to show now is that $T_{i+1}(z) = 2$ when $T_i(z) = 2$ (since the inequality is strict if $T_i(z) > 2$ and $T_i(z) \geq 2$ for all i). If $T_i(z) = 2$, then $T_{i+1}(z) = \lfloor \frac{2}{2} \rfloor + 1 = 2$.

To establish that $T_i(z)$ is weakly increasing in z for every i, the proof is by induction. For $a \leq b$, $T_1(a) = a \leq b = T_1(b)$. Now consider some $i > 1$ and suppose the induction hypothesis holds for $i-1$. Then $T_i(a) = \lfloor T_{i-1}(a)/2 \rfloor + 1 \leq \lfloor T_{i-1}(b)/2 \rfloor + 1 = T_i(b)$.

The sequence $\{n_j\}_{j=1}^{I(z)}$ defined by $n_j = T_{I(z)-j+1}(z)$—i.e., an inverted version of the sequence $\{T_i(z)\}_{i=1}^{I(z)}$—satisfies $n_{I(z)} = T_1(z) = z$, $n_1(z) = T_{I(z)} = 2$, and $n_{i-1} = T_{I(z)-(i-1)+1} = \lfloor \frac{T_{I(z)-(i-1)}}{2} \rfloor + 1 = \lfloor \frac{n_i}{2} \rfloor + 1$. Also, by the definition of $I(z)$, $T_i(z) > 2$ for any $i > I(z)$. So, if we can show that $I(z) = \lceil \log_2(z-1) \rceil + 1$, the proof is complete.

The proof of $I(z) = \lceil \log_2(z-1) \rceil + 1$ is as follows. Note that for $z = 2$, the sequence $\{z_i\}$ is simply $z_i = 2$ for all i which implies $I(2) = 1$. Since $\lceil \log_2(2-1) \rceil + 1 = 1$, the relationship holds for $z = 2$. So, now consider $z > 2$. The proof proceeds in the following steps. First, for the special $\{m_i\}$ sequence defined in Lemma 9, we show $T_j(m_i) = m_{i+1-j}$ for any $i \geq 1$ and any $j \leq i$. Second, we use this to show that $I(m_i) = i$ for all $i \geq 1$. Third, we show that $z \in (m_{i-1}, m_i]$ implies $I(z) = i$ by showing $I(m_i - 1) < I(z) \leq I(m_i)$. Fourth, we show that the i such that $z \in (m_{i-1}, m_i]$ is given by $\lceil \log_2(z-1) \rceil + 1$. This then gives $I(z) = \lceil \log_2(z-1) \rceil + 1$ since $I(z) = I(m_i) = i = \lceil \log_2(z-1) \rceil + 1$.

First, we show $T_j(m_i) = m_{i+1-j}$ for any $i \geq 1$ and any $j \leq i$. Fix some $i \geq 1$. The proof is by induction. For $j = 1$, $T_1(m_i) = m_i = m_{i+1-1}$. Now consider some j having $2 \leq j \leq i$ and suppose the induction hypothesis holds for $j-1$. Then

$$\begin{aligned} T_j(m_i) &= \lfloor \frac{T_{j-1}(m_i)}{2} \rfloor + 1 \\ &= \lfloor \frac{m_{i+1-(j-1)}}{2} \rfloor + 1 \\ &= \lfloor \frac{m_{i+2-j}}{2} \rfloor + 1 \\ &= \lfloor \frac{2m_{i+1-j} - 1}{2} \rfloor + 1 \\ &= m_{i+1-j}, \end{aligned}$$

which proves $T_j(m_i) = m_{i+1-j}$ for $j \leq i$. The fourth equality follows from the definition of $\{m_i\}$ in Lemma 9.

Second, we show $I(m_i) = i$ for all $i \geq 1$. Fix any $i \geq 1$. We just showed $T_j(m_i) = m_{i+1-j}$. Hence, $T_i(m_i) = m_1 = 2$ and $T_{i-1}(m_i) = m_2 = 3$. Consequently, the definition of I—which for a given z is defined as the smallest $i \geq 1$ such that $T_i(z) = 2$—gives $I(m_i) = i$ (recall T_j is decreasing in j).

Third, we show that $z \in (m_{i-1}, m_i]$ implies $I(z) = i$ by showing $I(m_i - 1) < I(z) \leq I(m_i)$. Note that, since $z > 2$ (having taken care of the $z = 2$ case already), there is some $i \geq 2$ such that $z \in (m_{i-1}, m_i]$ (since $m_1 = 2$). To see $I(z) \leq I(m_i)$, suppose not, that $I(z) > I(m_i)$. But then $2 = T_{I(z)}(z) < T_{I(m_i)}(z) \leq T_{I(m_i)}(m_i) = 2$, which is a contradiction. Therefore, $I(z) \leq I(m_i)$.

To see $I(m_{i-1}) < I(z)$, we begin by showing $T_j(m_{i-1}) < T_j(m_{i-1} + \varepsilon)$ for any $\varepsilon > 0$ and any $j \leq i-1$. Since $T_j(m_{i-1}) = m_{i-j}$, it is equivalent to show that $m_{i-j} < T_j(m_{i-1} + \varepsilon)$, which we

show by induction. Clearly, for $j = 1$, we have $m_{i-1} < m_{i-1} + \varepsilon = T_1(m_{i-1} + \varepsilon)$. Now consider $j > 1$ and suppose it is true for $j - 1$. Then

$$\begin{aligned} T_j(m_{i-1} + \varepsilon) &= \lfloor \frac{T_{j-1}(m_{i-1} + \varepsilon)}{2} \rfloor + 1 \\ &= \lfloor \frac{T_{j-1}(m_{i-1} + \varepsilon) - m_{i-j+1} + m_{i-j+1}}{2} \rfloor + 1 \\ &= \lfloor \frac{T_{j-1}(m_{i-1} + \varepsilon) - m_{i-j+1} + 2m_{i-j} - 1}{2} \rfloor + 1 \\ &= \lfloor \frac{T_{j-1}(m_{i-1} + \varepsilon) - m_{i-j+1} - 1}{2} \rfloor + m_{i-j} + 1 \end{aligned}$$

Now, since the induction hypothesis of $T_{j-1}(m_{i-1}+\varepsilon) > m_{i-j+1}$ gives $T_{j-1}(m_{i-1}+\varepsilon)-m_{i-j+1}-1 \geq 0$, one has

$$T_j(m_{i-1} + \varepsilon) \geq \lfloor \frac{0}{2} \rfloor + m_{i-j} + 1 = m_{i-j} + 1 > m_{i-j}.$$

Hence the proof by induction is complete.

Now, having established $T_j(m_{i-1}) < T_j(m_{i-1} + \varepsilon)$ for any $\varepsilon > 0$ and any $j \leq i - 1$, we show $I(m_{i-1}) < I(z)$. Suppose not, that $I(m_{i-1}) \geq I(z)$. Then since $z > m_{i-1}$, taking $\varepsilon = z - m_{i-1}$ we have $2 = T_{I(m_{i-1})}(m_{i-1}) < T_{I(m_{i-1})}(m_{i-1} + \varepsilon) = T_{I(m_{i-1})}(z) \leq T_{I(z)}(z) = 2$, which is a contradiction.

Lastly, we now show that the i such that $z \in (m_{i-1}, m_i]$ is given by $\lceil \log_2(z - 1) \rceil + 1$. That this holds can be seen as follows. Note that $z \in (m_{i-1}, m_i]$ implies $\log_2(z - 1) + 1 \in (\log_2(m_{i-1} - 1) + 1, \log_2(m_i - 1) + 1]$. Then, since $\log_2(m_j - 1) + 1 = j$ for all $j \geq 1$ (Lemma 9), we have $\log_2(z - 1) + 1 \in (i - 1, i]$. Then, by the definition of $\lceil \cdot \rceil$, one has $\lceil \log_2(z - 1) + 1 \rceil = i$, which of course is equivalent to $\lceil \log_2(z - 1) \rceil + 1 = i$.

We established the i such that $z \in (m_{i-1}, m_i]$ is $i = \lceil \log_2(z - 1) \rceil + 1$. Also we showed $i - 1 = I(m_{i-1}) < I(z) \leq I(m_i) = i$. Hence $I(z) = \lceil \log_2(z - 1) \rceil + 1$, which completes the proof. □

Now we give a more convenient bound than the one in Proposition 7. The main result will apply it with $\bar{\sigma}$ bounds corresponding to brute force and binary concavity.

Lemma 11. *Suppose $\bar{\sigma} : [1, \infty) \to \mathbb{R}^+$ is either the identity map ($\bar{\sigma}(\gamma) = \gamma$) or is a strictly increasing, strictly concave, and differentiable function. If $\bar{\sigma}(\gamma) \geq \sigma(\gamma)$ for all $\gamma \in \mathbb{Z}^{++}$, then an upper bound on function evaluations is*

$$3\bar{\sigma}(n') + \sum_{j=1}^{I-2} 2^j \bar{\sigma}(2^{-j}(n' - 1) + 1)$$

if $I > 2$ where $I = \lceil \log_2(n - 1) \rceil + 1$. An upper bound for $I \leq 2$ is $3\bar{\sigma}(n')$.

Proof. In keeping with the notation of the other proofs, let z and γ correspond to n and n', respectively. Fix some arbitrary $z \geq 2$. By Lemma 10, there is a strictly monotone increasing

sequence $\{z_i\}_{i=1}^I$ with $z_I = z$, $z_i = \lfloor \frac{z_{i+1}}{2} \rfloor + 1$ for $i < I$, and with $I = \lceil \log_2(z-1) \rceil + 1$ (and having $z_1 = 2$).

For $i > 1$ and any $\gamma \geq 1$, define

$$W(z_i, \gamma) := \max_{\gamma' \in \{1,\ldots,\gamma\}} M(z_{i-1}, \gamma') + M(z_{i-1}, \gamma - \gamma' + 1).$$

For $i = 1$, define $W(z_i, \cdot) = 0$. The definition of M gives $M(z_i, \gamma) = \sigma(\gamma) + W(z_i, \gamma)$ for any $i > 1$ with $M(z_1, \cdot) = 0$. Note that $W(z_2, \gamma) = W(z_1, \gamma) = 0$.

Define $\bar{W}$—which we will demonstrate is an upper bound and continuous version of W—as

$$\bar{W}(z_i, \gamma) := \bar{\sigma}^*(\gamma) + \max_{\gamma' \in [1,\gamma]} \bar{W}(z_{i-1}, \gamma') + \bar{W}(z_{i-1}, \gamma - \gamma' + 1)$$

for $i > 2$ with

$$\bar{\sigma}^*(\gamma) := \max_{\gamma' \in [1,\gamma]} \bar{\sigma}(\gamma') + \bar{\sigma}(\gamma - \gamma' + 1). \tag{28}$$

For $i = 1$ or 2, define $\bar{W}(z_i, \gamma) = 0$. Then $W(z_i, \gamma) \leq \bar{W}(z_i, \gamma)$ for all $i \geq 1$ and all $\gamma \in \mathbb{Z}^{++}$. The proof is by induction. They are equal for $i = 1$ and $i = 2$. Now consider an $i > 2$ and suppose it holds for $i - 1$. Then,

$$\begin{aligned}
W(z_i, \gamma) &= \max_{\gamma' \in \{1,\ldots,\gamma\}} M(z_{i-1}, \gamma') + M(z_{i-1}, \gamma - \gamma' + 1) \\
&= \max_{\gamma' \in \{1,\ldots,\gamma\}} \sigma(\gamma') + \sigma(\gamma - \gamma' + 1) + W(z_{i-1}, \gamma') + W(z_{i-1}, \gamma - \gamma' + 1) \\
&\leq \max_{\gamma' \in \{1,\ldots,\gamma\}} \sigma(\gamma') + \sigma(\gamma - \gamma' + 1) + \max_{\gamma' \in \{1,\ldots,\gamma\}} W(z_{i-1}, \gamma') + W(z_{i-1}, \gamma - \gamma' + 1) \\
&\leq \max_{\gamma' \in \{1,\ldots,\gamma\}} \bar{\sigma}(\gamma') + \bar{\sigma}(\gamma - \gamma' + 1) + \max_{\gamma' \in \{1,\ldots,\gamma\}} \bar{W}(z_{i-1}, \gamma') + \bar{W}(z_{i-1}, \gamma - \gamma' + 1) \\
&\leq \max_{\gamma' \in [1,\gamma]} \bar{\sigma}(\gamma') + \bar{\sigma}(\gamma - \gamma' + 1) + \max_{\gamma' \in [1,\gamma]} \bar{W}(z_{i-1}, \gamma') + \bar{W}(z_{i-1}, \gamma - \gamma' + 1) \\
&\leq \bar{\sigma}^*(\gamma) + \max_{\gamma' \in [1,\gamma]} \bar{W}(z_{i-1}, \gamma') + \bar{W}(z_{i-1}, \gamma - \gamma' + 1) \\
&= \bar{W}(z_i, \gamma).
\end{aligned}$$

If $\bar{\sigma}(x) = x$ for all x, then $\bar{\sigma}(\gamma') + \bar{\sigma}(\gamma - \gamma' + 1) = \gamma + 1$, which does not depend on γ'. So, $\bar{\sigma}^*(\gamma) = \gamma + 1 = 2\bar{\sigma}(\frac{\gamma+1}{2})$. If the $\bar{\sigma}$ function is strictly increasing, strictly concave, and differentiable, then the first order condition of the $\bar{\sigma}^*(\gamma)$ problem yields $\bar{\sigma}'(\gamma') = \bar{\sigma}'(\gamma - \gamma' + 1)$. The derivative is invertible (by strict concavity) and so $\gamma' = \frac{\gamma+1}{2}$. So, $\bar{\sigma}^*(\gamma) = \bar{\sigma}(\frac{\gamma+1}{2}) + \bar{\sigma}(\gamma - \frac{\gamma+1}{2} + 1)$, which gives $\bar{\sigma}^*(\gamma) = 2\bar{\sigma}(\frac{\gamma+1}{2})$, the same condition as in the linear case.[13] Hence,

$$\bar{\sigma}^*(\gamma) = 2\bar{\sigma}(\frac{\gamma+1}{2}). \tag{29}$$

[13] Since this is an interior solution and the problem is concave, the constraint $\gamma' \in [1, \gamma]$ is not binding.

So, for $i > 2$,

$$\bar{W}(z_i, \gamma) = 2\bar{\sigma}(\frac{\gamma+1}{2}) + \max_{\gamma' \in [1,\gamma]} \bar{W}(z_{i-1}, \gamma') + \bar{W}(z_{i-1}, \gamma - \gamma' + 1). \tag{30}$$

We will now show for $i > 2$ that $\bar{W}(z_i, \gamma) = 2\bar{\sigma}(\frac{\gamma+1}{2}) + 2\bar{W}(z_{i-1}, \frac{\gamma+1}{2})$, which gives a simple recursive relationship for the upper bound (for $i = 1$ or 2, $\bar{W}(z_i, \gamma) = 0$). First note that $2\bar{W}(z_i, \frac{\gamma+1}{2})$ is just $\bar{W}(z_i, \gamma') + \bar{W}(z_i, \gamma - \gamma' + 1)$ evaluated at $\frac{\gamma+1}{2}$, and so $\max_{\gamma' \in [1,\gamma]} \bar{W}(z_i, \gamma') + \bar{W}(z_i, \gamma - \gamma' + 1) \geq 2\bar{W}(z_i, \frac{\gamma+1}{2})$. So, it is sufficient to show $\max_{\gamma' \in [1,\gamma]} \bar{W}(z_i, \gamma') + \bar{W}(z_i, \gamma - \gamma' + 1) \leq 2\bar{W}(z_i, \frac{\gamma+1}{2})$. Now,

$$\begin{aligned}
&\max_{\gamma' \in [1,\gamma]} \bar{W}(z_i, \gamma') + \bar{W}(z_i, \gamma - \gamma' + 1) \\
&= \max_{\gamma' \in [1,\gamma]} 2\bar{\sigma}(\frac{\gamma'+1}{2}) + 2\bar{W}(z_{i-1}, \frac{\gamma'+1}{2}) + 2\bar{\sigma}(\frac{(\gamma - \gamma' + 1) + 1}{2}) + 2\bar{W}(z_{i-1}, \frac{(\gamma - \gamma' + 1) + 1}{2}) \\
&= 2 \max_{\gamma' \in [1,\gamma]} \bar{\sigma}(\frac{\gamma'+1}{2}) + \bar{\sigma}(\frac{\gamma+1}{2} - \frac{\gamma'+1}{2} + 1) + \bar{W}(z_{i-1}, \frac{\gamma'+1}{2}) + \bar{W}(z_{i-1}, \frac{\gamma+1}{2} - \frac{\gamma'+1}{2} + 1) \\
&= 2 \max_{\tilde{\gamma}' \in [1, \frac{\gamma+1}{2}]} \bar{\sigma}(\tilde{\gamma}') + \bar{\sigma}(\frac{\gamma+1}{2} - \tilde{\gamma}' + 1) + \bar{W}(z_{i-1}, \tilde{\gamma}') + \bar{W}(z_{i-1}, \frac{\gamma+1}{2} - \tilde{\gamma}' + 1) \\
&\leq 2 \max_{\tilde{\gamma}' \in [1, \frac{\gamma+1}{2}]} \bar{\sigma}(\tilde{\gamma}') + \bar{\sigma}(\frac{\gamma+1}{2} - \tilde{\gamma}' + 1) + 2 \max_{\tilde{\gamma}' \in [1, \frac{\gamma+1}{2}]} \bar{W}(z_{i-1}, \tilde{\gamma}') + \bar{W}(z_{i-1}, \frac{\gamma+1}{2} - \tilde{\gamma}' + 1) \\
&= 2 \max_{\tilde{\gamma}' \in [1, \frac{\gamma+1}{2}]} \bar{\sigma}(\tilde{\gamma}') + \bar{\sigma}(\frac{\gamma+1}{2} - \tilde{\gamma}' + 1) + 2 \left(\overline{W}(z_i, \frac{\gamma+1}{2}) - 2\overline{\sigma}(\frac{\frac{\gamma+1}{2}+1}{2}) \right) \\
&= 2\bar{\sigma}^*(\frac{\gamma+1}{2}) + 2\overline{W}(z_i, \frac{\gamma+1}{2}) - 4\overline{\sigma}(\frac{\frac{\gamma+1}{2}+1}{2}) \\
&= 4\bar{\sigma}(\frac{\frac{\gamma+1}{2}+1}{2}) + 2\overline{W}(z_i, \frac{\gamma+1}{2}) - 4\overline{\sigma}(\frac{\frac{\gamma+1}{2}+1}{2}) \\
&= 2\bar{W}(z_i, \frac{\gamma+1}{2})
\end{aligned}$$

The first equality follows from (30). The second equality follows from algebra. The third equality is just a change of variables where $\tilde{\gamma}' = (\gamma+1)/2$. The inequality follows from $\max(f+g) \leq \max f + \max g$ for any f, g. The fourth equality follows from evaluating (30) at $(\gamma+1)/2$ and manipulation. The fifth equality follows from the definition of $\bar{\sigma}^*$ in (28). The sixth equality follows from (29). The last equality simplifies. So, $\max_{\gamma' \in [1,\gamma]} \bar{W}(z_i, \gamma') + \bar{W}(z_i, \gamma - \gamma' + 1) = 2\bar{W}(z_i, \frac{\gamma+1}{2})$. Using this equality to replace the max in (30), one has (for $i > 2$) that

$$\bar{W}(z_i, \gamma) = 2\bar{\sigma}(\frac{\gamma+1}{2}) + 2\bar{W}(z_{i-1}, \frac{\gamma+1}{2}). \tag{31}$$

Now, fix any $\gamma \geq 1$ and define $\gamma_I := \gamma$ and $\gamma_i = \frac{\gamma_{i+1}+1}{2}$. Then $\gamma_i = 2^{i-I}(\gamma_I - 1) + 1$.[14] Then, for

[14]The proof is by induction. For $i = I$, $\gamma_I = 2^{I-I}(\gamma_I - 1) + 1 = \gamma_I$, which holds. Consider then some $i < I$, and suppose it holds for $i+1$. Then

$$\gamma_i = \frac{\gamma_{i+1}+1}{2} = \frac{(2^{i+1-I}(\gamma_I - 1) + 1) + 1}{2} = 2^{i-I}(\gamma_I - 1) + 1.$$

$i > 2$, (31) becomes

$$\bar{W}(z_i, \gamma_i) = 2\bar{\sigma}(\gamma_{i-1}) + 2\bar{W}(z_{i-1}, \gamma_{i-1}).$$

So, if $I > 2$, one can repeatedly expand the above to find a value for $\bar{W}(z_I, \gamma_I)$:

$$\begin{aligned}
\bar{W}(z_I, \gamma_I) &= 2\bar{\sigma}(\gamma_{I-1}) + 2\bar{W}(z_{I-1}, \gamma_{I-1}) \\
&= 2\bar{\sigma}(\gamma_{I-1}) + 2^2\bar{\sigma}(\gamma_{I-2}) + 2^2\bar{W}(z_{I-2}, \gamma_{I-2}) \\
&= 2\bar{\sigma}(\gamma_{I-1}) + \ldots + 2^{I-2}\bar{\sigma}(\gamma_2) + 2^{I-2}\bar{W}(z_2, \gamma_2) \\
&= 2\bar{\sigma}(\gamma_{I-1}) + \ldots + 2^{I-2}\bar{\sigma}(\gamma_2) \\
&= \sum_{j=1}^{I-2} 2^j \bar{\sigma}(\gamma_{I-j}) \\
&= \sum_{j=1}^{I-2} 2^j \bar{\sigma}(2^{I-j-I}(\gamma_I - 1) + 1) \\
&= \sum_{j=1}^{I-2} 2^j \bar{\sigma}(2^{-j}(\gamma_I - 1) + 1).
\end{aligned}$$

The first equalities are algebra, the fourth uses the definition of $\bar{W}(z_2, \cdot) = 0$, and the rest are algebra. If $I \leq 2$, then $\bar{W}(z_I, \gamma_I) = 0$.

Proposition 7 shows the number of required evaluations is less than or equal to $2\sigma(\gamma_I) + M(z_I, \gamma_I)$ for $z_I \geq 2$ and $\gamma_I \geq 1$. Since $M(z_i, \gamma) = \sigma(\gamma) + W(z_i, \gamma)$ for any $i > 1$ (with $M(z_1, \gamma) = 0$) and $W(z_i, \gamma) \leq \bar{W}(z_i, \gamma)$, the required function evaluations are weakly less than $3\sigma(\gamma_I) + \bar{W}(z_I, \gamma_I)$ for any I (recalling $W(z_I, \gamma_I) = 0$ for $I \leq 2$). Hence, if $I > 2$, then an upper bound is

$$3\sigma(\gamma_I) + \sum_{j=1}^{I-2} 2^j \bar{\sigma}(2^{-j}(\gamma_I - 1) + 1)$$

and if $I \leq 2$ an upper bound is $3\sigma(\gamma_I)$. □

Proposition 1 essentially applies the formula in Lemma 11 and simplifies. In the proof, there will be two partial sums, and we establish what they equal now.

Lemma 12. *For $r \neq 1$, $\sum_{j=a}^{b} r^j = (r^a - r^{b+1})/(1-r)$ and $\sum_{j=a}^{b} jr^j = \frac{ar^a - br^{b+1}}{1-r} + \frac{r^{a+1} - r^{b+1}}{(1-r)^2}$. For $r = 2$, $\sum_{j=a}^{b} r^j = 2^{b+1} - 2^a$ and $\sum_{j=a}^{b} jr^j = 2^a(2-a) + 2^{b+1}(b-1)$.*

Proof. The first sum, $\sum_{j=a}^{b} r^j$, is the standard geometric series and its sum can be compactly written as $(r^a - r^{b+1})/(1-r)$ for $r \neq 1$. For $r = 2$, this is $2^{b+1} - 2^a$.

The second sum, $\sum_{j=a}^{b} jr^j$, a sort of weighted geometric series, has no commonly known formula,

so we derive it. Define $S := \sum_{j=a}^{b} jr^j$. Then for $r \neq 1$,

$$\begin{aligned}
(1-r)S &= \sum_{j=a}^{b} jr^j - \sum_{j=a}^{b} jr^{j+1} \\
&= \sum_{j=a}^{b} jr^j - \sum_{j=a+1}^{b+1} (j-1)r^j \\
&= \sum_{j=a}^{b} jr^j - \sum_{j=a+1}^{b+1} jr^j + \sum_{j=a+1}^{b+1} r^j \\
&= \left(ar^a + \sum_{j=a+1}^{b} jr^j \right) - \left(\sum_{j=a+1}^{b} jr^j + (b+1)r^{b+1} \right) + \frac{r^{a+1} - r^{b+2}}{1-r}.
\end{aligned}$$

The first line is algebra, the second a change of indices, the third algebra, and the fourth separates out a term from each of the first two summations (and uses the geometric series formula to replace the third). Canceling the remaining summations, one then has

$$\begin{aligned}
(1-r)S &= ar^a - (b+1)r^{b+1} + \frac{r^{a+1} - r^{b+2}}{1-r} \\
&= ar^a - br^{b+1} - \frac{(1-r)r^{b+1}}{1-r} + \frac{r^{a+1} - r^{b+2}}{1-r} \\
&= ar^a - br^{b+1} + \frac{r^{a+1} - r^{b+1}}{1-r} \\
\Leftrightarrow S &= \frac{ar^a - br^{b+1}}{1-r} + \frac{r^{a+1} - r^{b+1}}{(1-r)^2}.
\end{aligned}$$

Plugging in $r = 2$ gives $S = b2^{b+1} - a2^a + 2^{a+1} - 2^{b+1} = 2^a(2-a) + 2^{b+1}(b-1)$. □

We can now prove Proposition 1 by applying Lemma 11 with $\overline{\sigma}$ bounds corresponding to brute force and binary concavity.

Proof of Proposition 1

Proof. From Lemma 11, the number of evaluations required by the algorithm is not greater than $3\bar{\sigma}(n') + \bar{W}(n, n')$ where $\bar{W}(n, n') := \sum_{j=1}^{I-2} 2^j \bar{\sigma}(2^{-j}(n'-1)+1)$ and $I = \lceil \log_2(n-1) \rceil + 1$. In the case of brute force, $\bar{\sigma}(\gamma) = \gamma$ is a valid upper bound on $\sigma(\gamma) = \gamma$. Plugging this into the expression for $\bar{W}(n, n')$, one has

$$\begin{aligned}
W(n, n') &= (I-2)(n'-1) + \sum_{j=1}^{I-2} 2^j \\
&= (I-2)(n'-1) + 2^{I-1} - 2 \\
&= (I-2)(n'-1) + 2^{\lceil \log_2(n-1) \rceil} - 2 \\
&\leq (I-2)(n'-1) + 2^{\log_2(n-1)+1} - 2
\end{aligned}$$

$$
\begin{aligned}
&\leq (I-2)(n'-1) + 2(n-1) - 2 \\
&= (\lceil \log_2(n-1) \rceil - 1)(n'-1) + 2n - 4 \\
&\leq (n'-1)\log_2(n-1) + 2n - 4.
\end{aligned}
$$

where we have used Lemma 12 to arrive at the second line. So, no more than $(n'-1)\log_2(n-1) + 3n' + 2n - 4$ evaluations are required.

In the case of binary concavity, Lemma 7 shows $\sigma(\gamma) = 2\lceil \log_2(\gamma) \rceil - 1$ for $\gamma \geq 3$ and $\sigma(\gamma) = \gamma$ for $\gamma \leq 3$ is an upper bound. Now consider $\bar{\sigma}(\gamma) = 2\log_2(\gamma) + 1$. It is a strictly increasing, strictly concave, and differentiable function. For $\gamma = 1$ or 2, one can plug in values to find $\sigma(\gamma) \leq \bar{\sigma}(\gamma)$. Additionally, for $\gamma \geq 3$, $\sigma(\gamma) \leq 2\lceil \log_2(\gamma) \rceil - 1 \leq 2(1 + \log_2(\gamma)) - 1 = \bar{\sigma}(\gamma)$ So, $\bar{\sigma}$ satisfies all the conditions of Lemma 11.

Plugging this $\bar{\sigma}$ into the bound, one finds

$$
\begin{aligned}
\bar{W}(n, n') &= \sum_{j=1}^{I-2} 2^j \left[1 + 2\log_2(2^{-j}(n'-1) + 1)\right] \\
&= (2^{I-1} - 2) + 2\sum_{j=1}^{I-2} 2^j \log_2(2^{-j}(n'-1) + 1)
\end{aligned}
$$

(using Lemma 12). To handle the $\log_2(2^{-j}(n'-1)+1)$ term, we break the summation into two parts, one with $2^{-j}(n'-1) < 1$ and one with $2^{-j}(n'-1) \geq 1$. We do this to exploit the following fact: For $x \geq 1$, $\log_2(x+1) \leq \log_2(x) + 1$ since they are equal at $x = 1$ the right hand side grows more quickly in x (i.e., the derivative of $\log_2(x+1)$ is less than the derivative of $\log_2(x) + 1$).

Let J be such that $j > J$ implies $2^{-j}(n'-1) < 1$ and $j \leq J$ implies $2^{-j}(n-1) \geq 1$. Then since $2^{-j}(n-1) = 1$ for $j = \log_2(n'-1)$, J is given by $\lfloor \log_2(n'-1) \rfloor$. Recall that in the statement of the proposition we assumed that $n' \geq 3$. So, $J \geq 1$.

$$
\begin{aligned}
\bar{W}(n, n') &= (2^{I-1} - 2) + 2\sum_{j=J+1}^{I-2} 2^j \log_2(\underbrace{2^{-j}(n'-1)}_{<1} + 1) + 2\sum_{j=1}^{J} 2^j \log_2(\underbrace{2^{-j}(n'-1)}_{\geq 1} + 1) \\
&\leq (2^{I-1} - 2) + 2\sum_{j=J+1}^{I-2} 2^j + 2\sum_{j=1}^{J} 2^j (1 + \log_2(2^{-j}(n'-1))) \\
&= (2^{I-1} - 2) + 2\sum_{j=1}^{I-2} 2^j + 2\sum_{j=1}^{J} 2^j \log_2(2^{-j}(n'-1)) \\
&= 3(2^{I-1} - 2) + 2\sum_{j=1}^{J} 2^j \log_2(2^{-j}(n'-1)) \\
&= 3(2^{I-1} - 2) + 2\sum_{j=1}^{J} 2^j (-j + \log_2(n'-1))
\end{aligned}
$$

$$= 3(2^{I-1} - 2) - 2\sum_{j=1}^{J} j2^j + 2\log_2(n'-1)\sum_{j=1}^{J} 2^j$$

The first line follows from the definition of J; the second from $\log_2(x+1) \leq 1 + \log_2(x)$ for $x \geq 1$; the third from algebra; the fourth from the standard geometric series formula; and the fifth and sixth from algebra. Then, using the weighted geometric sum found in Lemma 12, i.e., $\sum_a^b j2^j = 2^a(2-a) + 2^{b+1}(b-1)$,

$$\begin{aligned}
&= 3(2^{I-1} - 2) - 2(2^1(2-1) + 2^{J+1}(J-1)) + 2\log_2(n'-1)(2^{J+1} - 2) \\
&= 3(2^{I-1} - 2) - 4 - 2^{J+2}(J-1) + 2\log_2(n'-1)(2^{J+1} - 2) \\
&\leq 3(2^{I-1} - 2) - 4 - 2^{J+2}(J-1) + 2(J+1)(2^{J+1} - 2) \\
&= 3(2^{I-1} - 2) - 4 - 2^{J+2}(J-1) + (J+1)2^{J+2} - 4(J+1) \\
&= 3(2^{I-1} - 2) - 4 - J2^{J+2} + 2^{J+2} + J2^{J+2} + 2^{J+2} - 4(J+1) \\
&= 3(2^{I-1} - 2) - 4 + 2^{J+3} - 4J - 4 \\
&= 3 \cdot 2^{I-1} + 2^{J+3} - 4J - 14.
\end{aligned}$$

The first line applies the weighted geometric sum formula, the second simplifies, the third uses $\log_2(n'-1) \leq J+1$ (i.e., $\log_2(n'-1) \leq \lfloor \log_2(n'-1) \rfloor$), and the remaining lines use algebra.

Now, substituting the expressions for I and J,

$$\begin{aligned}
&= 3 \cdot 2^{\lceil \log_2(n-1) \rceil + 1 - 1} + 2^{\lfloor \log_2(n'-1) \rfloor + 3} - 4\lfloor \log_2(n'-1) \rfloor - 14 \\
&\leq 3 \cdot 2^{1 + \log_2(n-1)} + 2^{\log_2(n'-1)+3} - 4\lfloor \log_2(n'-1) \rfloor - 14 \\
&= 6(n-1) + 8(n'-1) - 4\lfloor \log_2(n'-1) \rfloor - 14 \\
&= 6n + 8n' - 4\lfloor \log_2(n'-1) \rfloor - 28 \\
&\leq 6n + 8n' - 4(\log_2(n'-1) - 1) - 28 \\
&= 6n + 8n' - 4\log_2(n'-1) - 24.
\end{aligned}$$

The above expression provides a bound for $\bar{W}(n, n')$. Hence, the total number of evaluations—which must be less than $3\bar{\sigma}(n') + \bar{W}(n, n')$—cannot exceed

$$\begin{aligned}
&3(2\log_2(n') + 1) + 6n + 8n' - 4\log_2(n'-1) - 24 \\
&= 6n + 8n' + 6\log_2(n') - 4\log_2(n'-1) - 21 \\
&\leq 6n + 8n' + 6(\log_2(n'-1) + 1) - 4\log_2(n'-1) - 21 \\
&= 6n + 8n' + 2\log_2(n'-1) - 15.
\end{aligned}$$

The second line is algebra, the third again uses $\log_2(x+1) \leq 1 + \log_2(x)$ for $x \geq 1$ (note $n' \geq 3$ in the statement of the proposition), and the last simplifies. □

F.3.3 Binary monotonicity in two dimensions

In proving the efficiency of the two-state binary monotonicity algorithm, we first establish a lemma.

Lemma 13. *Define* $m(a,b) = \lfloor \frac{a+b}{2} \rfloor$ *for* $b > a$ *with* $a, b \in \mathbb{Z}$. *Then* $m(a,b) - a + 1 \leq \lfloor \frac{b-a+1}{2} \rfloor + 1$ *and* $b - m(a,b) + 1 \leq \lfloor \frac{b-a+1}{2} \rfloor + 1$.

Proof. If exactly one of a, b is odd, then $m = \frac{b+a-1}{2}$; otherwise, $m = \frac{b+a}{2}$. So,

$$m - a + 1 \leq \frac{b+a}{2} - a + 1 = \frac{b-a+1}{2} + \frac{1}{2} \leq \lfloor \frac{b-a+1}{2} \rfloor + 1.$$

Now, take the case of exactly one of a, b being odd. Then $b - a$ is odd and $b - a + 1$ is even. In this case,

$$b - m + 1 = b - \frac{b+a-1}{2} + 1 = \frac{b-a+1}{2} + 1 = \lfloor \frac{b-a+1}{2} \rfloor + 1.$$

If on the other hand a, b are either both even or both odd, then $b - a + 1$ is odd. In this case,

$$b - m + 1 = b - \frac{b+a}{2} + 1 = \frac{b-a+1}{2} + \frac{1}{2} = \lfloor \frac{b-a+1}{2} \rfloor + 1.$$

□

We now give the proof of the two-state algorithm's cost bounds:

Proof of Proposition 2

Proof. Let an upper bound on the cost of the usual binary monotonicity algorithm given z states and γ choices as $C(z, \gamma)$. Then note that the cost of solving for $g(\cdot, 1)$ is less than $C(n_1, n')$, as is the cost of solving for $g(\cdot, n_2)$.

Let $g(\cdot, \cdot; \pi)$ give the policy selected by the algorithm when the objective function is π. Let $T^{exact}(\underline{j}, \overline{j}; \pi)$ denote the exact cost of the algorithm for recovering $g(\cdot, j; \pi)$ for all $j \in \{\underline{j}+1, \ldots, \overline{j}-1\}$ when the objective function is π and $g(\cdot, \underline{j}; \pi)$ and $g(\cdot, \overline{j}; \pi)$ are known. (That is, the cost from Step 3 onward). Define for $z > 2$ and $\gamma \geq 1$

$$T^*(z, \gamma) = \sup_{\underline{j}, \overline{j} \in \{1, \ldots, n_2\}, \pi} C(n_1, \gamma) + T^{exact}(\underline{j}, m(\underline{j}, \overline{j}); \pi) + T^{exact}(m(\underline{j}, \overline{j}), \overline{j}; \pi)$$
$$\text{s.t. } \overline{j} - \underline{j} + 1 \leq z, \;\; \underline{j} < \overline{j}$$
$$g(n_1, \overline{j}; \pi) - g(1, \underline{j}; \pi) + 1 \leq \gamma$$

where $m(a,b)$ is the integer midpoint $\lfloor \frac{a+b}{2} \rfloor$. For $z = 2$, define $T^*(z, \cdot) = 0$. Note $T^*(z, \gamma)$ must be weakly increasing in both arguments.

Fix some $(\underline{j}, \overline{j}, \pi)$ with $\underline{j}, \overline{j} \in \{1, \ldots, n_2\}$ and $\underline{j} < \overline{j}$, and note that $(\underline{j}, \overline{j}, \pi)$ is in the choice set of the $T^*(\overline{j} - \underline{j} + 1, g(n_1, \overline{j}; \pi) - g(1, \underline{j}; \pi) + 1)$ problem. We now show T^{exact} is bounded by T^* in that $T^{exact}(\underline{j}, \overline{j}; \pi) \leq T^*(\overline{j} - \underline{j} + 1, g(n_1, \overline{j}; \pi) - g(1, \underline{j}; \pi) + 1)$. To see this, note that if $\overline{j} - \underline{j} + 1 = 2$ then $T^{exact}(\underline{j}, \overline{j}; \pi) = 0$, in which case $T^*(\overline{j} - \underline{j} + 1, \cdot) = 0$ by definition. On the other hand, if $\overline{j} - \underline{j} + 1 > 2$,

then $T^{exact}(\underline{j},\overline{j};\pi) \le C(n_1, g(n_1,\overline{j};\pi) - g(1,\underline{j};\pi)+1) + T^{exact}(\underline{j}, m(\underline{j},\overline{j});\pi) + T^{exact}(m(\underline{j},\overline{j}),\overline{j};\pi)$ because C bounds the cost of the one-dimensional algorithm. Comparing with the definition of T^*, $T^*(\overline{j}-\underline{j}+1, g(n_1,\overline{j};\pi)-g(1,\underline{j};\pi)+1)$ is necessarily larger because $(\underline{j},\overline{j},\pi)$ is in its choice set.

Now, using this bound and the definition of T^* gives

$$
\begin{aligned}
&T^*(z,\gamma)\\
&\le \sup_{\underline{j},\overline{j},\pi} C(n_1,\gamma) + T^*(m-\underline{j}+1, \gamma'(\pi,\underline{j},\overline{j})) + T^*(\overline{j}-m+1, g(n_1,\overline{j};\pi)-g(1,m;\pi)+1)\\
&= \sup_{\underline{j},\overline{j},\pi} C(n_1,\gamma) + T^*(m-\underline{j}+1, \gamma'(\pi,\underline{j},\overline{j})) + T^*\begin{pmatrix}\overline{j}-m+1, & g(n_1,\overline{j};\pi)-g(1,\underline{j};\pi)+1\\ & +g(1,\underline{j};\pi)-g(1,m;\pi)\end{pmatrix}\\
&\le \sup_{\underline{j},\overline{j},\pi} C(n_1,\gamma) + T^*(m-\underline{j}+1, \gamma'(\pi,\underline{j},\overline{j})) + T^*(\overline{j}-m+1, \gamma+g(1,\underline{j};\pi)-g(1,m;\pi))\\
&= \sup_{\underline{j},\overline{j},\pi} C(n_1,\gamma) + T^*(m-\underline{j}+1, \gamma'(\pi,\underline{j},\overline{j}))\\
&\qquad + T^*(\overline{j}-m+1, \gamma+g(n_1,m;\pi)-g(1,m;\pi)+1-\gamma'(\pi,\underline{j},\overline{j}))
\end{aligned}
$$

where $m = m(\underline{j},\overline{j})$ and we have defined $\gamma'(\pi,\underline{j},\overline{j}) := g(n_1,m;\pi) - g(1,\underline{j};\pi)+1$. The first relation follows from T^{exact} being less than T^*, the second from adding and subtracting $g(1,\underline{j};\pi)$, the third from the constraint that $g(n_1,\overline{j};\pi)-g(1,\underline{j};\pi)+1 \le \gamma$ and T^* being increasing in its second argument, and the last from adding and subtracting γ' and manipulation. With this definition of γ', the constraint $g(n_1,\overline{j};\pi)-g(1,\underline{j};\pi)+1 \le \gamma$ is equivalent to $\gamma'(\pi,\underline{j},\overline{j}) \le \gamma$. So, $\gamma'(\pi,\underline{j},\overline{j}) \in [1,\gamma]$.

Using our λ-based restriction, we have as an implication of it that $g(n_1,m;\pi)-g(1,m;\pi)+1 \le \lambda(g(n_1,\overline{j};\pi)-g(1,\underline{j};\pi)+1) \le \lambda\gamma$. So, because T^* is increasing in its second argument,

$$T^*(z,\gamma) \le \sup_{\underline{j},\overline{j},\pi} C(n_1,\gamma) + T^*(m-\underline{j}+1, \gamma'(\pi,\underline{j},\overline{j})) + T^*(\overline{j}-m+1, (1+\lambda)\gamma - \gamma'(\pi,\underline{j},\overline{j}))$$

Note now that π only shows up in γ'. Since the choice set implies $\gamma' \in [1,\gamma]$, we can drop π and allow $\gamma' \in [1,\gamma]$ to be chosen directly:

$$T^*(z,\gamma) \le \sup_{\underline{j},\overline{j},\gamma' \in [1,\gamma]} C(n_1,\gamma) + T^*(m-\underline{j}+1,\gamma') + T^*(\overline{j}-m+1, (1+\lambda)\gamma-\gamma')$$

By Lemma 13, $m-\underline{j}+1$ and $\overline{j}-m+1$ are less than $\lfloor \frac{\overline{j}-\underline{j}+1}{2} \rfloor + 1$ which—because of the constraint $\overline{j}-\underline{j}+1 \le z$—must be less than $\lfloor \frac{z}{2} \rfloor + 1$. So,

$$T^*(z,\gamma) \le \sup_{\underline{j},\overline{j},\gamma' \in [1,\gamma]} C(n_1,\gamma) + T^*(\lfloor \frac{z}{2} \rfloor + 1, \gamma') + T^*(\lfloor \frac{z}{2} \rfloor + 1, (1+\lambda)\gamma - \gamma'),$$

which no longer depends on $\underline{j}$ or $\overline{j}$. Hence, dropping these from the choice set, one has

$$T^*(z,\gamma) \leq C(n_1,\gamma) + \sup_{\gamma' \in [1,\gamma]} T^*(\lfloor \frac{z}{2} \rfloor + 1, \gamma') + T^*(\lfloor \frac{z}{2} \rfloor + 1, (1+\lambda)\gamma - \gamma').$$

Defining $\overline{T}(z,\gamma) := C(n_1,\gamma) + \max_{\gamma' \in [1\gamma]} \overline{T}(\lfloor \frac{z}{2} \rfloor + 1, \gamma') + \overline{T}(\lfloor \frac{z}{2} \rfloor + 1, (1+\lambda)\gamma - \gamma')$ for $z > 2$ and 0 otherwise, we then have $\overline{T}(z,\gamma)$ as an upper bound for $T^*(z,\gamma)$ (by induction with $T^*(2,\gamma) = \overline{T}(2,\gamma) = 0$ as the base case).

Now, by Lemma 10, there is a unique sequence $\{z_i\}_{i=1}^I$ with $z_I = z$, $\lfloor \frac{z_i}{2} \rfloor + 1 = z_{i-1}$, $z_1 = 2$, and $z_i > 2$ for all $i > 1$ and $I = \lceil \log_2(z-1) \rceil + 1$. Then,

$$\overline{T}(z_i,\gamma) = C(n_1,\gamma) + \sup_{\gamma' \in [1,\gamma]} \overline{T}(z_{i-1},\gamma') + \overline{T}(z_{i-1},(1+\lambda)\gamma - \gamma')$$

for all i.

For binary monotonicity with brute force grid search, the upper bound on function counts in Proposition 1 is linear increasing in γ. So suppose C is linear increasing in γ. In that case, we will show $\overline{T}(z_i,\gamma)$ is linear increasing in γ for all $i \geq 2$ and that

$$\overline{T}(z_i,\gamma) = C(n_1,\gamma) + 2\overline{T}(z_{i-1}, \gamma\frac{(1+\lambda)}{2}).$$

First note this is trivially the case for $i = 2$ since in that case $\overline{T}(z_{i-1},\cdot) = \overline{T}(z_1,\cdot) = \overline{T}(2,\cdot) = 0$. Now, suppose that it holds for $i-1$. Then continuity gives that the maximum is attained so sup can be replaced with max. Because of linearity, $\overline{T}(z_{i-1},\gamma') + \overline{T}(z_{i-1},(1+\lambda)\gamma - \gamma')$ is independent of γ'. Consequently, $\max_{\gamma'} \overline{T}(z_{i-1},\gamma') + \overline{T}(z_{i-1},(1+\lambda)\gamma - \gamma') = 2\overline{T}(z_{i-1}, \gamma\frac{(1+\lambda)}{2})$. Hence, $\overline{T}(z_i,\gamma)$ will also be linearly increasing and satisfy the recursive formulation above.

Now, expanding the recursive formulation and defining $c := (1+\lambda)/2$,

$$\begin{aligned}
\overline{T}(z_I,\gamma) &= C(n_1,\gamma) + 2\overline{T}(z_{I-1},\gamma c) \\
&= C(n_1,\gamma) + 2(C(n_1,\gamma c) + 2\overline{T}(z_{I-2},(\gamma c)c) \\
&= C(n_1,\gamma) + 2C(n_1,\gamma c) + 2^2\overline{T}(z_{I-2},\gamma c^2) \\
&= C(n_1,\gamma) + 2C(n_1,\gamma c) + \ldots + 2^{I-2}C(n_1,\gamma c^{I-2}) + 2^{I-1}\overline{T}(z_{I-(I-1)},\gamma c^{I-1}) \\
&= C(n_1,\gamma) + 2C(n_1,\gamma c) + \ldots + 2^{I-2}C(n_1,\gamma c^{I-2}) + 2^{I-1}\overline{T}(2,\gamma c^{I-1}) \\
&= C(n_1,\gamma) + 2C(n_1,\gamma c) + \ldots + 2^{I-2}C(n_1,\gamma c^{I-2}) \\
&= \sum_{i=0}^{I-2} 2^i C(n_1,\gamma c^i)
\end{aligned}$$

Plugging in $c = (1+\lambda)/2$, $z_I = n_2$ (corresponding to the first time Step 3 is reached), and $\gamma = n'$

(corresponding to Step 3 being reached with the worst case $g(\cdot,1)=1$ and $g(\cdot,n_2)=n'$),

$$\overline{T}(n_2,n') = \sum_{i=0}^{I-2} 2^i C(n_1, 2^{-i}(1+\lambda)^i n')$$

where $I = \lceil \log_2(n_2-1) \rceil + 1$.

With brute force grid search, a valid C is $C(n_1,\gamma) = 2n_1 + \gamma(\log_2(n_1)+3)$. Then

$$\begin{aligned}
\overline{T}(n_2,n') &= \sum_{i=0}^{I-2} 2^i \left(2n_1 + \left(2^{-i}(1+\lambda)^i n'\right)\left(\log_2(n_1)+3\right)\right) \\
&= n'(\log_2(n_1)+3)\sum_{i=0}^{I-2}(1+\lambda)^i + 2n_1\sum_{i=0}^{I-2}2^i \\
&= n'(\log_2(n_1)+3)\frac{1-(1+\lambda)^{I-1}}{-\lambda} + 2n_1(2^{I-1}-1) \\
&= n'(\log_2(n_1)+3)\frac{(1+\lambda)^{I-1}-1}{\lambda} + 2n_1(2^{I-1}-1)
\end{aligned}$$

using the formulas in Lemma 12.

Now, $I = \lceil \log_2(n_2-1)\rceil + 1$ implies $I-1 \le \log_2(n_2-1)+1$. So, defining $\kappa := \log_2(1+\lambda)$ so that $(1+\lambda) = 2^\kappa$,

$$\begin{aligned}
\overline{T}(n_2,n') &= n'(\log_2(n_1)+3)\frac{2^{\kappa(I-1)}-1}{\lambda} + 2n_1(2^{I-1}-1) \\
&\le n'(\log_2(n_1)+3)\lambda^{-1}2^{\kappa(I-1)} + 2n_1(2^{I-1}-1) \\
&\le n'(\log_2(n_1)+3)\lambda^{-1}2^{\kappa(\log_2(n_2-1)+1)} + 2n_1(2^{\log_2(n_2-1)+1}-1) \\
&= n'(\log_2(n_1)+3)\lambda^{-1}(2^{\log_2(n_2-1)})^\kappa 2^\kappa + 2n_1(2(n_2-1)-1) \\
&= n'(\log_2(n_1)+3)\lambda^{-1}(n_2-1)^\kappa 2^\kappa + 4n_1n_2 - 6n_1 \\
&\le n'(\log_2(n_1)+3)\lambda^{-1}n_2^\kappa(1+\lambda) + 4n_1n_2 - 6n_1 \\
&= (1+\lambda)\lambda^{-1}\log_2(n_1)n'n_2^\kappa + 3(1+\lambda)\lambda^{-1}n'n_2^\kappa + 4n_1n_2 - 6n_1 \\
&= (\lambda^{-1}+1)\log_2(n_1)n'n_2^\kappa + 3(\lambda^{-1}+1)n'n_2^\kappa + 4n_1n_2 - 6n_1.
\end{aligned}$$

This bound does *not* include the cost of solving for $g(\cdot,1)$ and $g(\cdot,n_2)$ using the standard binary algorithm. Including this cost, the algorithm's total cost is less than

$$\begin{aligned}
&2C(n_1,n') + \overline{T}(n_2,n') \\
&\le 2n'(\log_2(n_1)+3) + 4n_1 + (1+\lambda^{-1})\log_2(n_1)n'n_2^\kappa + 3(1+\lambda^{-1})n'n_2^\kappa + 4n_1n_2 - 6n_1 \\
&\le (1+\lambda^{-1})\log_2(n_1)n'n_2^\kappa + 3(1+\lambda^{-1})n'n_2^\kappa + 4n_1n_2 + 2n'\log_2(n_1) + 6n',
\end{aligned}$$

which is the bound stated in the proposition.

Now, suppose that $\lambda < 1$ provides a uniform bound on $(g(n_1,j) - g(1,j) + 1)/(g(n_1,j+1) -$

$g(1, j-1)+1)$ (this also implies $\lambda > 0$). So, $\kappa \in (0,1)$ (since $\kappa = \log_2(1+\lambda)$). To characterize the algorithm's $O(n_1 n_2)$ behavior when $n_1 = n' =: n$ and $n_2 = \rho^{-1} n$, divide the bound by $n_1 n_2 = n^2/\rho$ to arrive at

$$c\frac{\log_2(n) n^{1+\kappa}}{n^2} + 4 + \frac{o(n^2)}{n^2}$$

where c is a positive constant. Then it is enough to show that $\log(n) n^{1+\kappa}$ (using that natural log has the same asymptotics as $\log_2$) grows more slowly than n^2. The ratio $\log(n) n^{1+\kappa}/n^2$ equals $\log(n)/n^{1-\kappa}$. Using l'Hopital's rule as $n \to \infty$, if the limit exists it is the same as the limit of $(1/n)/((1-\kappa)n^{-\kappa}) = n^{-1+\kappa}/(1-\kappa)$. This equals 0 since $\kappa \in (0,1)$. Consequently, the cost is $O(n_1 n_2)$ with a hidden constant of 4. □

References

C. Arellano. Default risk and income fluctuations in emerging economies. *American Economic Review*, 98(3):690–712, 2008.

Y. Bai and J. Zhang. Financial integration and international risk sharing. *Journal of International Economics*, 86(1):17–32, 2012.

B. Heer and A. Maußner. *Dynamic General Equilibrium Modeling: Computational Methods and Applications*. Springer, Berlin, Germany, 2005.

H. A. Hopenhayn and E. C. Prescott. Stochastic monotonicity and stationary distributions for dynamic economies. *Econometrica*, 60(6):1387–1406, 1992.

P. Milgrom and I. Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70(2):583–601, 2002.

D. Simchi-Levi, X. Chen, and J. Bramel. *The Logic of Logistics: Theory, Algorithms, and Applications for Logistics Management (Third Edition)*. Springer, New York, 2014.

D. M. Topkis. Minimizing a submodular function on a lattice. *Operations Research*, 26(2):305–321, 1978.

D. M. Topkis. *Supermodularity and complementarity*. Princeton University Press, Princeton, N.J., 1998.