Potential games, path independence and Poisson's binomial distribution

Sen, Debapriya

Ryerson University

2 February 2018

Potential games, path independence and Poisson's binomial distribution

Debapriya Sen*

February 2, 2018

Abstract

This paper provides a simple characterization of potential games in terms of path independence. Using this characterization we propose an algorithm to determine if a finite game is potential or not. We define the storage requirement for our algorithm and provide its upper bound. The number of equations required in this algorithm is lower or equal to the number obtained in the algorithms proposed in the recent literature. We also show that for games with same numbers of players and strategy profiles, the number of equations for our algorithm is maximum when all players have the same number of strategies. To obtain our results, the key technique of this paper is to identify an associated Poisson's binomial distribution. This distribution enables us to derive explicit forms of the number of equations, storage requirement and related aspects.

Keywords: potential games; zero strategy; path independence; Poisson's binomial distribution; storage requirement

*Ryerson University, Toronto, Ontario, Canada

1 Introduction

A potential of a game is a function of its strategy profiles such that if a profile is obtained from another through a unilateral deviation by one player, the difference in the potential between these profiles equals the gain in payoff of the deviating player. If a game admits a potential, it is called a potential game. Introduced by Monderer and Shapley (1996), potential games have been applied to study diverse issues such as coordination (Anderson et al., 2001; Bramoullé, 2007), congestion (Chien and Sinclair, 2011), networks (Roughgarden and Tardos, 2002; Bramoullé et al. 2014), consensus problems (Marden et al., 2009) and wireless systems like cognitive radio (Neel et al., 2004; Ellingsæter et al., 2012).

This paper provides a simple characterization of potential games and proposes an algorithm to determine whether a finite game is potential or not. The key technique of this paper is to use properties of a statistical distribution, which enables us to get explicit forms of number of equations of our algorithm, as well as comparable algorithms from the recent literature.

Consider any game with a finite number n players, each having finitely many strategies. For every player, we denote one of its strategy as the "zero strategy" and call the rest "positive strategies". Corresponding to any such game, there is an associated *Poisson's binomial distribution* (the distribution that corresponds to the number of successes in n independent but not necessarily identical Bernoulli trials), where the probability of success in the i-th trial is the proportion of positive strategies of player i.

A strategy profile $\widetilde{s}$ is called a predecessor of a profile s if s is obtained from $\widetilde{s}$ via a unilateral deviation of a player from its zero strategy to some positive strategy. We construct a weighted directed graph from the game, whose vertices are the strategy profiles. The profile z with all zero strategies act as the origin. For any predecessor-successor pair $(\widetilde{s}, s)$, a directed edge is drawn from $\widetilde{s}$ to s and the weight assigned to this edge is the gain in payoff of the corresponding deviating player. The length of any directed path in this graph is the sum of weights of all edges that appear in the path. For any profile s, the *path independence* property holds if all paths from the origin z to s have the same length. We show that a game admits a potential if and only if the path independence property holds for all vertices of its associated graph (Theorem 1).

Based on the characterization above, we construct an algorithm to determine whether a game is potential or not. Using the expectation and success probabilities of the associated Poisson's binomial distribution, the number of equations required for this algorithm can be explicitly stated (Theorem 2).

Among the papers of the existing literature, Sandholm (2010), Hino (2011) and Cheng et al. (2016) are closely related to our work. Using the dimensions of subspaces of different classes of games, Cheng et al. (2016) find the same number of equations as our algorithm to detect a potential game. Using a similar reasoning Sandholm (2010) also finds the same number in the special case where all players in a game have the same number of strategies. Our key point of distinction with the existing literature is the method of using the properties of the associated Poisson's binomial distribution. Hino (2011) proposes an alternative algorithm,[1] but does not give an explicit form of its number of equations. Using the Poisson's

[1]In the special case when all players have the same number of strategies, the number of equations in the algorithm of Hino (2011) coincides with the number obtained in Theorem 3.5 of Sandholm (2010).

binomial distribution, we determine the number of equations in Hino's algorithm and show that it is always at least as large as the number of equations required in our algorithm. Furthermore, for games in which there are at least three players with two or more strategies, our algorithm results in lower number of equations (Corollary 1).

The existing literature (e.g., Sandholm, 2010; Hino 2011) discusses two measures to determine computational burden for an algorithm: (i) the number of equations needed for its execution and (ii) its storage requirement. However, the literature does not have a precise definition of storage requirement. We give a definition of storage requirement for our algorithm and find its upper bound. To define storage requirement we note that four types of objects are needed to execute our algorithm: vertices, weights associated with vertices, edges and weights associated with edges. The storage required at any step of the algorithm is simply the total number of objects at that step. The storage requirement of the algorithm is the maximum required storage over all of its steps. Using the Poisson's binomial distribution we are able to provide an upper bound to the storage requirement of our algorithm (Theorem 3). This bound is based on a remarkable result of Darroch (1964), who shows that the mode of a Poisson's binomial distribution differs from its mean by at most 1.

Finally we address another issue that has not received much attention in the literature. We ask whether having same or different numbers of strategies across players increases the number of equations of our algorithm. To address this question we begin with the situation where all players have the same number of strategies. Then we alter the numbers of strategies of players keeping the number of strategy profiles the same. It is shown that the number of equations for our algorithm is maximum when all players have the same number of strategies (Theorem 4). This conclusion is in part driven by the general result that for different Poisson's binomial distributions with the same mean, the corresponding binomial distribution (the one where the probability of success stays the same across trials) results in the maximum variance (see, e.g., Hoeffding, 1956; Wang, 1993; Pitman, 1997).

One specific application of our algorithm can be in situations where the problem is to "design" a potential game. For instance, consider the problem of frequency allocation in a wireless system (see, e.g., Ellingsæter et al., 2012). In this problem the players are wireless access points and the set of strategies of each player is its available channels. It is often useful to design this interaction as a potential game. This is an example where given players and strategy profiles, one has to assign payoffs to ensure that the game has a potential function. Our algorithm and the related results can be useful for such problems.

The paper is organized as follows. We present the model, together with the associated Poisson's binomial distribution, in Section 2. The characterization of potential games in terms of path independence and our algorithm are presented in Section 3. We discuss other aspects of the algorithm such as storage requirement in Section 4. We conclude in Section 5. Some proofs are presented in the Appendix.

2 The model

Let $\Gamma = \langle N, (S_i)_{i\in N}, (u_i)_{i\in N}\rangle$ be a game in strategic form with a finite number of players, with each player having a finite number of strategies. Let $N = \{1, \ldots, n\}$ be the set of players where $n \geq 2$. For $i \in N$, let S_i be the set of strategies of player i. Let $S = \times_{i\in N} S_i$ be the set of strategy profiles and denote $S_{-i} = \times_{j\neq i} S_j$. Let $u_i : S \to \mathbb{R}$ be the payoff function

of player i.

Player i has $k_i \geq 1$ strategies, i.e., $|S_i| = k_i$. It will be useful for our analysis to denote $S_i = \{0, 1, \ldots, k_i - 1\}$. Thus each player i has a distinct *zero strategy* ($s_i = 0$) and possibly other *positive strategies* ($s_i > 0$).

Definition 1 (Monderer and Shapley, 1996) The game Γ is a *potential game* if there is a function $P : S \to \mathbb{R}$ (called a *potential function* of Γ) such that for every $i \in N$, $s_i, \widetilde{s}_i \in S_i$ and $s_{-i} \in S_{-i}$

$$u_i(s_i, s_{-i}) - u_i(\widetilde{s}_i, s_{-i}) = P(s_i, s_{-i}) - P(\widetilde{s}_i, s_{-i}) \tag{1}$$

We begin with the following simple observation. The proof is straightforward and hence omitted.

Lemma 1 *The game Γ is a potential game if and only if there is a function $P : S \to \mathbb{R}$ such that for every $i \in N$, $s_i \in S_i$ and $s_{-i} \in S_{-i}$*

$$u_i(s_i, s_{-i}) - u_i(0, s_{-i}) = P(s_i, s_{-i}) - P(0, s_{-i}) \tag{2}$$

2.1 A classification of strategy profiles

In light of Lemma 1, the zero strategy of a player will play a central role in our analysis to determine whether a game is potential or not. Denote the strategy profile where every player plays its zero strategy by z, i.e.,

$$z \equiv (0, \ldots, 0) \tag{3}$$

It will be useful to classify the strategy profiles according to the number of zero strategies in a profile. Specifically, for $t = 0, 1, \ldots n$, define

$$V_t := \{s \in S | s \text{ has } t \text{ positive strategies}\} \tag{4}$$

Thus, V_t is the set of all strategy profiles where $n - t$ players play their zero strategies and the remaining t players play positive strategies. Note that $V_t \cap V_{t'} = \emptyset$ for $t \neq t'$ and any strategy profile s is an element of V_t for some $t = 0, 1, \ldots, n$, so we can partition $S = \cup_{t=0}^{n} V_t$. Also observe that V_0 is the singleton set $\{z\}$.

2.2 An associated probability distribution

For $i = 1, \ldots, n$, let p_i be the proportion of positive strategies for player i and let $q_i \equiv 1 - p_i$, that is,

$$p_i = (k_i - 1)/k_i, q_i = 1/k_i \text{ so that } k_i = 1/q_i \tag{5}$$

Since $k_i \geq 1$, we have $0 \leq p_i < 1$. Let $Y \equiv Y(n; p_1, \ldots, p_n)$ denote the total number of successes in n independent but not necessarily identical Bernouilli trials, where the probability of success of the i-th trial is p_i. Then Y follows a *Poisson's binomial* (or *Poisson-binomial*) distribution. We know that (see, e.g., Wang, 1993):

$$E(Y) = \sum_{i=1}^{n} p_i \text{ and } \text{Var}(Y) = \sum_{i=1}^{n} p_i q_i \tag{6}$$

From the definition of V_t in (4), observe that

$$\Pr(Y = t) = \frac{|V_t|}{\sum_{\tau=0}^{n} |V_\tau|} \text{ for } t = 0, 1, \ldots, n \tag{7}$$

and hence $E(Y) = \sum_{t=0}^{n} t\Pr(Y = t) = \sum_{t=0}^{n} t|V_t| / \sum_{t=0}^{n} |V_t|$. From (6), we have

$$E(Y) = \frac{\sum_{t=0}^{n} t|V_t|}{\sum_{t=0}^{n} |V_t|} = \sum_{i=1}^{n} p_i \tag{8}$$

Since $\sum_{t=0}^{n} |V_t|$ corresponds to the total number of strategy profiles of the game Γ, using (5) we have

$$\sum_{t=0}^{n} |V_t| = \prod_{i=1}^{n} k_i = \frac{1}{\prod_{i=1}^{n} q_i} \tag{9}$$

2.3 A directed graph representation of Γ

Definition 2 Let $s \in V_t$. A strategy profile $\widetilde{s} \in V_{t-1}$ is called a *predecessor* of s (and s a *successor* of $\widetilde{s}$) if there is a player i such that (a) $s_i > 0$ and $\widetilde{s}_i = 0$ and (b) $s_j = \widetilde{s}_j$ for all $j \neq i$. The player i for which (a) and (b) hold is called the *critical player* of the pair $(\widetilde{s}, s)$.

Since there are t players who play positive strategies in $s \in V_t$, any such s has exactly t predecessors, each having a different critical player. In particular, z has no predecessor and any $s \in V_1$ has only one predecessor z.

By Definition 2, if $\widetilde{s}$ is a predecessor of s, then $s = (s_i, s_{-i})$ and $\widetilde{s} = (0, s_{-i})$ with $s_i > 0$, where i is the critical player of $(\widetilde{s}, s)$. The profile s can be reached from profile $\widetilde{s}$ if player i makes a unilateral deviation from its zero strategy to the strategy s_i. For any predecessor-successor pair $(\widetilde{s}, s)$ with critical player i, denote

$$\Delta(\widetilde{s}, s) := u_i(s) - u_i(\widetilde{s}) = u_i(s_i, s_{-i}) - u_i(0, s_{-i}) \tag{10}$$

Thus, $\Delta(\widetilde{s}, s)$ presents the gain in payoff of the critical player from its unilateral deviation when we move from $\widetilde{s}$ to s.

Consider condition (2) of Lemma 1. Note that if $s_i = 0$, this condition holds for any function P, so let $s_i > 0$. Using Definition 2 and (10) in Lemma 1, it follows that a function $P : S \to \mathbb{R}$ is a potential function if and only if *for every* predecessor-successor pair $(\widetilde{s}, s)$,

$$P(s) - P(\widetilde{s}) = \Delta(\widetilde{s}, s) \tag{11}$$

Motivated by (11), we construct the weighted directed graph $G(\Gamma)$ from the game Γ as follows.

(i) Present each $s \in S = \cup_{t=0}^{n} V_t$ as a vertex.

(ii) For any predecessor-successor pair $(\widetilde{s}, s)$, draw a directed edge which originates at $\widetilde{s}$ and terminates at s. Denote this edge by $(\widetilde{s}, s)$.

(iii) Put the weight $\Delta(\widetilde{s}, s)$ (given by (10)) on the edge $(\widetilde{s}, s)$.

Total number of edges in G: Let e_t denote the number of edges that terminate at some vertex of V_t. Since every vertex of V_t has t predecessors, we have $e_t = t|V_t|$. Therefore the total number of edges in G is $\eta(G) = \sum_{t=0}^{n} e_t = \sum_{t=0}^{n} t|V_t|$. By (8) and (9), $\eta(G)$ can be presented in terms of the expectation and the success probabilities of the Poisson's binomial distribution of Section 2.2 as:

$$\eta(G) = E(Y) \sum_{t=0}^{n} |V_t| = \frac{\sum_{i=1}^{n} p_i}{\prod_{i=1}^{n} q_i} \tag{12}$$

Definition 3 A sequence $\gamma = (y_0, y_1, \ldots, y_m)$ is a *directed path* of $G(\Gamma)$ if for all ℓ, y_ℓ is a predecessor of $y_{\ell+1}$. The vertex y_0 is called the *origin* and y_m the *terminus* of the path γ. The number of edges in the path γ is m, given by $(y_0, y_1), \ldots, (y_{m-1}, y_m)$. The *length* of the path γ is the sum of weights over these m edges, given by

$$L(\gamma) := \Delta(y_0, y_1) + \Delta(y_1, y_2) + \ldots + \Delta(y_{m-1}, y_m) = \sum_{\ell=0}^{m-1} \Delta(y_\ell, y_{\ell+1}) \tag{13}$$

If $m = 0$, the path $\gamma = (y_0)$ has only one vertex and no edge, so its length is zero.

One immediate property of directed paths can be noted. By (13) it follows that for two directed paths $\gamma = (y_0, \ldots, y_m)$ and $\gamma' = (y_0, \ldots, y_{m-1})$,

$$L(\gamma) = L(\gamma') + \Delta(y_{m-1}, y_m) \tag{14}$$

2.4 Illustrative examples

It will be useful to illustrate the directed graph presentation with the help of examples.

Example 1 Consider a three-firm Cournot oligopoly game Γ_1 where quantities supplied by firms are restricted to be non-negative integers. This can correspond to a situation where firms produce an indivisible good that can be only sold in integers units. Specifically the set of players is the set of three firms $N = \{1, 2, 3\}$. To make the game finite, assume further that each player is capacity constrained: players $1, 2$ can produce at most 1 unit whereas player 3 can produce at most 2 units. So we have $S_1 = \{0, 1\}$, $S_2 = \{0, 1\}$ and $S_3 = \{0, 1, 2\}$. Players simultaneously choose their quantities. Given (s_1, s_2, s_3), the market price is given by the equation $F(s_1, s_2, s_3) = 6 - (s_1 + s_2 + s_3)$. Assume that all players have zero cost of production, so the payoff of a player is simply its revenue, given by

$$u_i(s_1, s_2, s_3) = F(s_1, s_2, s_3)s_i = [6 - (s_1 + s_2 + s_3)]s_i \tag{15}$$

Define the function $P : S \to \mathbb{R}$ as

$$P(s_1, s_2, s_3) = 6 \sum_{i=1}^{3} s_i - \sum_{i=1}^{3} s_i^2 - \sum_{1 \le i < j \le 3} s_i s_j \tag{16}$$

It can be verified that the function P is a potential function[2] for the game Γ_1.

[2]See Monderer and Shapley (1996) for potential functions of Cournot oligopoly games with more general demand and cost functions. For an analysis of Cournot oligopoly games in integers, see Todd (2016).

Note that for the game Γ_1, each player i has a "zero strategy" which corresponds to $s_i = 0$. Each of players $1, 2$ has one positive strategy, whereas player 3 has two positive strategies. Using the definition of V_t from (4), for this game we have $V_0 = \{(0,0,0)\}$, $V_1 = \{(1,0,0),(0,1,0),(0,0,1),(0,0,2)\}$, $V_2 = \{(1,1,0),(0,1,1),(1,0,1),(1,0,2),(0,1,2)\}$ and $V_3 = \{(1,1,1),(1,1,2)\}$.

Diagram 1 has the directed graph presentation of Γ_1. At the top is the sole vertex of V_0: $(0,0,0)$. At next level are vertices of V_1 and so on. Between any predecessor-successor pair, there is a directed edge. For this game we have $p_i = (k_i - 1)/k_i = 1/2$ for $i = 1, 2$ and $p_3 = (k_3 - 1)/k_3 = 2/3$, so that $\sum_{i=1}^{3} p_i = 5/3$. Since $q_i = 1 - p_i$, we have $\prod_{i=1}^{3} q_i = 1/12$. Hence $(\sum_{i=1}^{3} p_i)/\prod_{i=1}^{3} q_i = 20$, so by (12) it follows that the directed graph has 20 edges in total.

The number inside the box of each edge gives the weight of that edge. For example, consider the edge between $(0,0,0)$ and $(1,0,0)$. For this edge, the critical player is player 1 (as we reach from $(0,0,0)$ to $(1,0,0)$ through the unilateral deviation of player 1 from its zero strategy to $s_1 = 1$). So the weight on this edge is $u_1(1,0,0) - u_1(0,0,0) = 5 - 0 = 5$. Similarly consider the edge between $(1,1,0)$ and $(1,1,2)$. For this edge, the critical player is player 3 and weight on this edge is $u_3(1,1,2) - u_3(1,1,0) = 4 - 0 = 4$.

Example 2 Consider a three-player minimum-effort coordination game Γ_2 where[3] the set of players is $N = \{1,2,3\}$. Each player i chooses an effort level s_i. The effort levels of players act as perfect complements to determine the output. Specifically the output is given by $\min\{s_1, s_2, s_3\}$. For any player i, the cost of effort level s_i is cs_i where c is a positive constant. To make the game finite, assume that $S_1 = \{0,1\}$, $S_2 = \{0,1\}$ and $S_3 = \{0,1,2\}$. Given (s_1, s_2, s_3), the payoff of a player is output net of its cost, so we have

$$u_i(s_1, s_2, s_3) = \min\{s_1, s_2, s_3\} - cs_i \tag{17}$$

Define the function $P : S \to \mathbb{R}$ as

$$P(s_1, s_2, s_3) = \min\{s_1, s_2, s_3\} - c\sum_{i=1}^{3} s_i \tag{18}$$

It can be verified that the function P is a potential function for the game Γ_2.

Note that for $t = 0, 1, 2, 3$, the set V_t for this game is the same as in Example 1. Diagram 2 has the directed graph presentation of the game Γ_2. As before, between any predecessor-successor pair there is a directed edge. By (12), the total number of edges for this graph is $(\sum_{i=1}^{3} p_i)/\prod_{i=1}^{3} q_i = 20$.

The number inside the box of each edge gives the weight of that edge. For example, consider the edge between $(0,0,0)$ and $(1,0,0)$. For this edge, the critical player is player 1. So the weight on this edge is $u_1(1,0,0) - u_1(0,0,0) = -c - 0 = -c$.

3 A characterization and an algorithm for potential games

Based on the directed graph representation of the last section, we define the path independence property that is used to characterize potential games. Based on this characterization,

[3]See Anderson et al. (2001) for general properties of such games.

we propose an algorithm to determine if a game is potential or not.

3.1 Path independence

For a finite game Γ consider the set V_t defined in (4). Let $s \in V_t$ for some $t \geq 0$. It will be useful for our analysis to consider all directed paths with origin at z and terminus at s. Since the number of players with positive strategies at $s \in V_t$ is t, to reach s from z, we need t unilateral deviations. As these deviations can occur in $t!$ different ways, this is the number of directed paths with origin z and terminus s. Each of these paths has exactly t edges.

Definition 4 The *path independence* property (PI) holds for $s \in S$ if all directed paths with origin z and terminus s have the same length.

Theorem 1 *The game Γ is a potential game if and only if the path independence property holds for all $s \in S$.*

Proof The "if part": Suppose PI holds for all $s \in S$. Then all directed paths with origin z and terminus s have the same length. Denote this length by $\lambda(s)$. Define the function $P : S \to \mathbb{R}$ as

$$P(s) := \lambda(s) \tag{19}$$

Observe in particular that $P(z) = \lambda(z) = 0$. We prove that P defined in (19) is a potential function of Γ by showing that P satisfies (11) for any predecessor-successor pair $(\widetilde{s}, s)$.

First suppose $s \in V_1$. Then the only predecessor of s is z. Denote $\Delta(z, s) := u_i(s) - u_i(z)$. There is only one path with origin z and terminus s and this path has length $\Delta(z, s)$. So for this case $P(s) = \lambda(s) = \Delta(z, s)$. Since $P(z) = 0$, we have $P(s) - P(z) = \Delta(z, s) = u_i(s) - u_i(z)$, so (11) holds.

Next suppose $s \in V_t$ for some $t \geq 2$. Consider any predecessor $\widetilde{s} \in V_{t-1}$ of s. Let $\gamma = (y_0, \ldots, y_{t-1}, y_t)$ be a directed path with origin z and terminus s that passes through $\widetilde{s}$, i.e., $y_0 = z$, $y_{t-1} = \widetilde{s}$ and $y_t = s$. Consider the path $\widetilde{\gamma} = (y_0, \ldots, y_{t-1})$ with origin z and terminus $\widetilde{s}$. Then it follows by (14) that

$$L(\gamma) = L(\widetilde{\gamma}) + \Delta(y_{t-1}, y_t) = L(\widetilde{\gamma}) + \Delta(\widetilde{s}, s) \tag{20}$$

Since PI holds for both s and $\widetilde{s}$, we have $L(\gamma) = \lambda(s) = P(s)$ and $L(\widetilde{\gamma}) = \lambda(\widetilde{s}) = P(\widetilde{s})$. Using this in (20), we have $P(s) - P(\widetilde{s}) = \Delta(\widetilde{s}, s)$, so (11) holds. This completes the proof of the "if part".

The "only if part": Suppose Γ is a potential game. Then there is a function P such that (11) holds for any predecessor-successor pair $(\widetilde{s}, s)$. By (13), the length of any directed path $\gamma = (y_0, \ldots, y_m)$ is $L(\gamma) = \sum_{\ell=0}^{m-1} \Delta(y_\ell, y_{\ell+1})$. Since $(y_\ell, y_{\ell+1})$ is a predecessor-successor pair, by (11), we have $\Delta(y_\ell, y_{\ell+1}) = P(y_{\ell+1}) - P(y_\ell)$ for all ℓ, so that $L(\gamma) = \sum_{\ell=0}^{m-1} [P(y_{\ell+1}) - P(y_\ell)] = P(y_m) - P(y_0)$. Taking $y_0 = z$ and $y_m = s$, all paths with origin z and terminus s has the same length $P(s) - P(z)$, proving that PI holds for $s \in S$. ■

Remark 1 It can be seen from Diagrams 1,2 that for each of the two graphs, path independence holds all vertices. For instance, in Diagram 1 consider the vertex $s = (1, 0, 2)$. There are two paths from $z = (0, 0, 0)$ to s: (i) $(0, 0, 0) \to (1, 0, 0) \to (1, 0, 2)$ and (ii) $(0, 0, 0) \to (0, 0, 2) \to (1, 0, 2)$. The length of path (i) is $5 + 6 = 11$ and the length of path (ii) is $8 + 3 = 11$.

3.2 An algorithm to determine a potential game

Based on the characterization of Theorem 1, the next theorem identifies the number of equations required to verify PI for all $s \in S$. This number is also represented in terms of the expectation and success probabilities of the Poisson's binomial distribution of Section 2.2.

Theorem 2 *Let $p_i = (k_i - 1)/k_i$, $q_i = 1/k_i$ and let $Y \equiv Y(n; p_1, \ldots, p_n)$ denote the total number of successes in n independent Bernouilli trials, where the probability of success of the i-th trial is p_i. Denote*

$$A_n(k_1, \ldots, k_n) := \sum_{t=2}^{n} (t-1)|V_t| \tag{21}$$

By checking $A_n(k_1, \ldots, k_n)$ equations, it can be determined whether Γ is a potential game. Moreover

$$A_n(k_1, \ldots, k_n) = 1 + [E(Y) - 1] \sum_{t=0}^{n} |V_t| = 1 + \frac{\sum_{i=1}^{n} p_i - 1}{\prod_{i=1}^{n} q_i} \tag{22}$$

Proof Using Theorem 1, we determine whether Γ is a potential game or not by checking if PI holds for all $s \in S = \cup_{t=0}^{n} V_t$. Note that PI always holds for all $s \in V_0 \cup V_1$. We check the property recursively as follows.

Let $t \geq 2$. Suppose PI holds for all $s \in V_\ell$ for $\ell = 0, 1, \ldots, t-1$. Consider any $s \in V_t$. Any directed path with origin z and terminus s must pass through some predecessor $\widetilde{s} \in V_{t-1}$ of s. Consider two such paths $\gamma = (y_0, \ldots, y_{t-1}, y_t)$ and $\gamma' = (y'_0, \ldots, y'_{t-1}, y'_t)$ that pass through the same predecessor $\widetilde{s}$, i.e., $y_0 = y'_0 = z$, $y_t = y'_t = s$ and $y_{t-1} = y'_{t-1} = \widetilde{s}$. Denoting $\gamma_1 = (y_0, \ldots, y_{t-1})$ and $\gamma'_1 = (y'_0, \ldots, y'_{t-1})$, observe that

$$L(\gamma) = L(\gamma_1) + \Delta(\widetilde{s}, s) \text{ and } L(\gamma') = L(\gamma'_1) + \Delta(\widetilde{s}, s) \tag{23}$$

Note that both γ_1 and γ'_1 have origin z and terminus $\widetilde{s}$. Since $\widetilde{s} \in V_{t-1}$, PI holds for $\widetilde{s}$, so we have $L(\gamma_1) = L(\gamma'_1)$. Denote this common length by $\lambda(\widetilde{s})$. Then it follows from (23) that all directed paths with origin z and terminus s that pass through the same predecessor $\widetilde{s}$ have the same length $\lambda(\widetilde{s}) + \Delta(\widetilde{s}, s)$. Since $s \in V_t$, the vertex s has exactly t predecessors. Denoting these predecessors by $\widetilde{s}^1, \ldots, \widetilde{s}^t$, it follows that PI holds for s if and only if

$$\lambda(\widetilde{s}^1) + \Delta(\widetilde{s}^1, s) = \ldots = \lambda(\widetilde{s}^t) + \Delta(\widetilde{s}^t, s)$$

This implies that we need to check $t-1$ equations for every $s \in V_t$. Therefore the number of equations we have to check to see if PI holds for all $s \in V_t$ is $(t-1)|V_t|$. Applying this recursive argument for $t = 2, \ldots, n$, the total number of equations we need to check to see if PI holds for all $s \in S$ is $\sum_{t=2}^{n}(t-1)|V_t|$. Noting that $|V_0| = 1$, (22) follows by using (8) and (9) in (21). ∎

Remark 2 Note that for each of the games Γ_1, Γ_2 in Examples 1,2, we have $n = 3$, $|V_2| = 5$, $|V_3| = 2$ (see Diagrams 1,2). Each vertex in V_2 has 2 edges terminating at it, so we need to check $2 - 1 = 1$ equation for each such vertex. Similarly each vertex in V_3 has 3 edges terminating at it, so we need to check $3 - 1 = 2$ equations for each such vertex. So the total number of equations to check, as in (21), is $A_n(k_1, k_2, k_3) = A_3(2, 2, 3) = \sum_{t=2}^{3}(t-1)|V_t| = (1 \times |V_2|) + (2 \times |V_3|) = (1 \times 5) + (2 \times 2) = 9$. Also note that for each of the games Γ_1, Γ_2, we have $p_i = (k_i - 1)/k_i = 1/2$ for $i = 1, 2$ and $p_3 = (k_3 - 1)/k_3 = 2/3$. So $\sum_{i=1}^{3} p_i - 1 = 2/3$ and $\prod_{i=1}^{3} q_i = 1/12$. Hence $1 + (\sum_{i=1}^{3} p_i - 1)/\prod_{i=1}^{3} q_i = 1 + (2/3)12 = 9$, so (22) holds.

3.2.1 Comparison with recent literature

Consider as before a finite game Γ with set of players $N = \{1, \ldots, n\}$, where player $i \in N$ has k_i strategies. In the algorithm proposed in the recent paper of Hino (2011) (see also Theorem 3.5, Sandholm, 2010), the number of equations required to verify whether Γ is a potential game is given as follows, where $i, j, \ell \in N$:

$$B_n(k_1, \ldots, k_n) = \sum_{1 \leq i < j \leq n} \theta_{ij} \text{ where } \theta_{ij} := (k_i - 1)(k_j - 1) \prod_{\ell \neq i,j} k_\ell \tag{24}$$

Corollary 1 compares this with the number of equations required for our algorithm.

Corollary 1

(i)

$$B_n(k_1, \ldots, k_n) = \sum_{t=2}^{n} \binom{t}{2} |V_t| \tag{25}$$

Moreover

$$B_n(k_1, \ldots, k_n) = \frac{1}{2}\left[\text{Var}(Y) + (E(Y))^2 - E(Y)\right] \sum_{t=0}^{n} |V_t| = \frac{\sum_{1 \leq i < j \leq n} p_i p_j}{\prod_{i=1}^{n} q_i} \tag{26}$$

where p_i, q_i, Y are as in Theorem 2.

(ii) *$B_2(k_1, k_2) = A_2(k_1, k_2)$ and for $n \geq 3$, $B_n(k_1, \ldots, k_n) \geq A_n(k_1, \ldots, k_n)$ with strict inequality if at least three players have two or more strategies.*

Proof See the Appendix. ■

Using the Poisson's binomial distribution of Section 2.2, Corollary 1 identifies the number of equations required for the algorithm proposed by Hino (2011). It is also shown that this number is always at least as large as the number of equations required in our algorithm. Furthermore, for games in which there are at least three players with two or more strategies, our algorithm requires a lower number of equations.

4 Other aspects of algorithm

In this section we look at two other aspects of our algorithm: (i) storage requirement and (ii) same versus different numbers of strategies across players.

4.1 Storage requirement

As mentioned earlier, the literature discusses two measures to determine the computational burden for an algorithm: (a) the number of equations needed and (b) its storage requirement. However, a precise definition of storage requirement is lacking in the literature. In this section we define storage requirement for our algorithm and provide its upper bound.

Four types of objects are needed to execute our algorithm: (i) vertices, (ii) weights associated with vertices, (iii) edges and (iv) weights associated with edges. Consider any

step τ of the algorithm. Denote by $N_\tau(v)$ = number of vertices, $N_\tau(w_v)$ = number of weights associated with vertices, $N_\tau(e)$ = number of edges and $N_\tau(w_e)$ = number of weights associated with edges to run step τ of the algorithm.

Definition 5 The *storage requirement at step* τ (SR_τ) of the algorithm is the total number of objets needed to run step τ of the algorithm, that is, SR_τ is the sum $N_\tau(v) + N_\tau(w_v) + N_\tau(e) + N_\tau(w_e)$.

Definition 6 The *storage requirement for the algorithm* is the maximum of SR_τ over all steps τ of the algorithm.

Consider the Poisson's binomial variable Y of Section 2.2. The variable Y is used in the next theorem to give an upper bound to the storage requirement of our algorithm, where $\lfloor x \rfloor$ stands for the largest integer not exceeding x.

Theorem 3 *For* $\lambda = 0, 1$, *let*

$$\beta_\lambda := \frac{4 \Pr(Y = \lfloor \sum_{i=1}^n p_i \rfloor + \lambda)}{\prod_{i=1}^n q_i} + 2n - 1 \tag{27}$$

where p_i, q_i, Y *are as in Theorem* 2. *The storage requirement for the algorithm of Theorem* 2 *is bounded above by either* β_0 *or* β_1.

Proof See the Appendix. ∎

Recall that for the Poisson's binomial variable Y, we have $E(Y) = \sum_{i=1}^n p_i$. Darroch (1964) has shown that the mode (that is, the most probable number of successes) of a Poisson's binomial distribution differs from its mean by at most 1. Specifically, for a Poisson's binomial variable Y, the mode is either $\lfloor E(Y) \rfloor = \lfloor \sum_{i=1}^n p_i \rfloor$ or $\lfloor E(Y) \rfloor + 1 = \lfloor \sum_{i=1}^n p_i \rfloor + 1$ or both.[4] Theorem 3 shows that the probability of this mode determines an upper bound of the storage requirement of our algorithm.

4.2 Same versus different numbers of strategies for players

In this section we ask whether having same or different numbers of strategies across players increases the number of equations for our algorithm. Our approach is as follows. Fix the number of players n and begin with the case where all players have the same number k of strategies so that the number of strategy profiles is k^n. Now alter the numbers of strategies across players keeping the number of strategy profiles the same, that is, consider n-tuples $(k_1, \ldots, k_n)$ keeping $\prod_{i=1}^n k_i = k^n$. Theorem 4 shows that $A_n(k_1, \ldots, k_n)$, $B_n(k_1, \ldots, k_n)$ are both maximum when $k_1 = \ldots = k_n = k$.

Theorem 4 *Let* $n \geq 2$ *and* $k \geq 1$. *The following hold for all* n*-tuples of positive integers* $(k_1, \ldots, k_n)$ *such that* $\prod_{i=1}^n k_i = k^n$.

(i) $A_n(k_1, \ldots, k_n) \leq A_n(k, \ldots, k)$ *with equality iff* $k_1 = \ldots = k_n = k$.

(ii) $B_n(k_1, \ldots, k_n) \leq B_n(k, \ldots, k)$ *with equality iff* $k_1 = \ldots = k_n = k$.

(iii) *If either* $k_i \geq 2$ *for at least four* i, *or* $k_i \geq 3$ *for at least three* i, *then* $B_n(k_1, \ldots, k_n) - A_n(k_1, \ldots, k_n) \leq B_n(k, \ldots, k) - A_n(k, \ldots, k)$ *with equality iff* $k_1 = \ldots = k_n = k$.

[4] See Theorem 4 (p.1321) of Darroch (1964) for the complete description of the mode. See also Pitman (1997, p.284).

Proof The results trivially hold for $k = 1$, so consider $k \geq 2$. Let $(k_1, \ldots, k_n)$ be any n-tuple of positive integers such that $\prod_{i=1}^{n} k_i = k^n$. Since $q_i = 1/k_i$, we have $\prod_{i=1}^{n} q_i = 1/k^n$. The arithmetic mean-geometric mean inequality implies that for any $r > 0$:

$$\sum_{i=1}^{n} q_i^r \geq n \prod_{i=1}^{n} q_i^{r/n} = \frac{n}{k^r} \text{ with equality iff } q_1 = \ldots = q_n = \frac{1}{k} \tag{28}$$

(i) As $q_i = 1 - p_i$, from (22), we have

$$A_n(k_1, \ldots, k_n) = 1 + k^n(n-1) - k^n \sum_{i=1}^{n} q_i \tag{29}$$

The result of (i) follows by applying (28) with $r = 1$ for the last term of (29).

(ii) Since $\sum_{1 \leq i < j \leq n} p_i p_j = [(\sum_{i=1}^{n} p_i)^2 - \sum_{i=1}^{n} p_i^2]/2$ and $q_i = 1 - p_i$, from (26), we have

$$B_n(k_1, \ldots, k_n) = \frac{k^n}{2} \left[\left(n - \sum_{i=1}^{n} q_i \right)^2 - \sum_{i=1}^{n} (1 - q_i)^2 \right] = \frac{k^n}{2} \left[f\left(\sum_{i=1}^{n} q_i \right) - \sum_{i=1}^{n} q_i^2 \right] \tag{30}$$

where $f(x) := n - 1 + [x - (n-1)]^2$.

If $(k_1, \ldots, k_n)$ is such that $k_i = k^n$ for some i and $k_j = 1$ for all $j \neq i$ (that is, all players except one have only one strategy), taking $q_i = 1/k^n$ and $q_j = 1$ for all $j \neq i$ it follows from (30) that $B_n(k_1, \ldots, k_n) = 0 < B_n(k, \ldots, k) = n(n-1)k^{n-2}(k-1)^2/2$ (since $n, k \geq 2$).

To complete the proof, consider $(k_1, \ldots, k_n)$ such that $k_i \geq 2$ for at least two i (that is, at least two players have 2 or more strategies). In that case, $q_i \leq 1/2$ for at least two i and hence $\sum_{i=1}^{n} q_i \leq (1/2) + (1/2) + (n-2) = n - 1$. Note that f is decreasing in $\sum_{i=1}^{n} q_i$ (this is because $\sum_{i=1}^{n} q_i \leq n - 1$ and $f(x)$ is decreasing for $x \leq n - 1$). Using this, the result follows by applying (28) with $r = 1$ for the first term and $r = 2$ for the second term of (30).

(iii) Denote $g(x) := n - 2 + [x - (n-2)]^2$. Note from (29) and (30) that

$$B_n(k_1, \ldots, k_n) - A_n(k_1, \ldots, k_n) = \frac{k^n}{2} \left[g\left(\sum_{i=1}^{n} q_i \right) - \sum_{i=1}^{n} q_i^2 \right] - 1 \tag{31}$$

If $k_i \geq 2$ for at least four i (that is, at least four players have 2 or more strategies), then $\sum_{i=1}^{n} q_i \leq 4(1/2) + (n-4) = n - 2$. If $k_i \geq 3$ for at least three i (that is, at least three players have 3 or more strategies), then $\sum_{i=1}^{n} q_i \leq 3(1/3) + (n-3) = n - 2$. Thus, in either case $\sum_{i=1}^{n} q_i \leq n - 2$. Since $g(x)$ is decreasing for $x \leq n - 2$, we conclude that g is decreasing in $\sum_{i=1}^{n} q_i$. Using this, the result follows by applying (28) with $r = 1$ for the first term and $r = 2$ for the second term of (30). ■

Theorem 4 shows that beginning from a situation where all players have the same number of strategies, the number of equations required for both our algorithm and the algorithm of Hino (2011) falls when the number of strategies across players are altered keeping the number of strategy profiles the same. Furthermore, provided enough players have two or more strategies, the difference between the numbers of equations of these two algorithms also falls when players have different numbers of strategies.

The results of Theorem 4 can be useful in problems of designing a potential game. As mentioned in the introduction, consider a wireless system problem (see, e.g., Ellingsæter et al., 2012) in which there are 4 access points (players), each having 2 channels (strategies). So the number of strategy profiles is $2^4 = 16$. Note that one can add players with only one strategy without affecting the interaction of a game (moreover, since $k_i = 1$ implies $p_i = 0$ and $q_i = 1$, by (22), adding players with only one strategy does not alter the number of equations for our algorithm). Given this observation and noting that $8 \times 2 \times 1 \times 1 = 16$, it follows by Theorem 4 that designing a potential game with 2 access points where one point has 8 channels and the other one has 2 will require lower number of equations compared to the case of 4 access points in which each one has 2 channels. Thus given the same number of strategy profiles, the number of equations is lower when there are few players having many strategies rather than many players having the same number of strategies.

To a certain extent, the results of Theorem 4 are driven by the general property that among different Poisson's binomial distributions with the same mean, the corresponding binomial distribution has the maximum variance. For instance, discussing the inequalities of Hoeffding (1956), Pitman (1997, p.283) states that among all Poisson's binomial distributions "...on $\{0, 1, \ldots, n\}$ with a given mean μ, the binomial (n, p) distribution for $p = \mu/n$ is the one that is "most spread out."" A precise statement of this result is presented in Theorem 1 of Wang (1993, p.301). Discussing this result, Wang (1993, p.302) points out that "...to estimate an unknown proportion $\overline{p}$, *the unbiased sample mean from a sequence of non-identically distributed Bernoulli random variables has smaller variance than the uniformly minimum variance unbiased estimate obtained by using the binomial density b with parameter* $\overline{p}$." (italics in the original)

5 Concluding remarks

This paper presents a simple characterization of potential games in terms of path independence. Based on this characterization we propose an algorithm to determine whether a finite game is potential or not. In terms of the number of required equations, our algorithm does at least as good or better than the algorithms proposed in the recent literature. In particular, for games where at least three players have two or more strategies, our algorithm requires a lower number of equations. We give a precise definition of the storage requirement and using the result of Darroch (1964), provide an upper bound of the storage requirement for our algorithm. We also address the question of whether the number of equations required for our algorithm increases with same or different numbers of strategies across players.

The key contribution of this paper is the method of using a statistical distribution. Identifying an associated Poisson's binomial distribution with any finite game, we use the properties of this distribution to derive our results. This distribution enables us to derive an explicit form for the number of equations of our algorithm and helps us to better understand related aspects such as the storage requirement. Our approach can be specifically useful for problems where payoffs are assigned to design a strategic interaction as a potential game.

Appendix

Proof of Corollary 1 (i) Since a player $i \in N$ has $k_i - 1$ positive strategies and k_i strategies in total, the term θ_{ij} defined in (24) corresponds to the number of strategy profiles where both players i, j play positive strategies. Let θ_{ij}^t be the number of strategy profiles in V_t where both i, j play positive strategies. Then $\theta_{ij} = \sum_{t=0}^{n} \theta_{ij}^t$. Since $\theta_{ij}^0 = \theta_{ij}^1 = 0$, we have $\theta_{ij} = \sum_{t=2}^{n} \theta_{ij}^t$. Using this in (24), we have

$$B_n(k_1, \ldots, k_n) = \sum_{1 \le i < j \le n} \theta_{ij} = \sum_{1 \le i < j \le n} \sum_{t=2}^{n} \theta_{ij}^t = \sum_{t=2}^{n} \sum_{1 \le i < j \le n} \theta_{ij}^t \tag{32}$$

Fix any $t \in \{2, \ldots, n\}$ and consider the sum $\psi_t := \sum_{1 \le i < j \le n} \theta_{ij}^t$. For any strategy profile $s \in V_t$, exactly t players play positive strategies. Since out of these t players, two players $i < j$ can be chosen in $\binom{t}{2}$ ways, it follows that the number of times a profile $s \in V_t$ is counted in this sum is $\binom{t}{2}$. Therefore $\psi_t = \binom{t}{2}|V_t|$. Using this in (32), the result in (25) follows.

Since $\mathrm{Var}(Y) = E(Y^2) - (E(Y))^2$, the first equality of (26) follows by using (7) in (25). Since $q_i = 1 - p_i$, the second equality follows from (6) and (9) by noting that $(\sum_{i=1}^{n} p_i)^2 - \sum_{i=1}^{n} p_i^2 = 2 \sum_{1 \le i < j \le n} p_i p_j$.

(ii) We give two alternative proofs of (ii).

First alternative proof: We compare $A_n(k_1, \ldots, k_n)$ from (21) with $B_n(k_1, \ldots, k_n)$ from (25). The first part follows by noting that $\binom{t}{2} - (t-1) = 0$ if $t = 2$. For the second part, note that $\binom{t}{2} - (t-1) = \binom{t-1}{2}$ if $t \ge 3$. So for $n \ge 3$ we have $B_n(k_1, \ldots, k_n) - A_n(k_1, \ldots, k_n) = \sum_{t=3}^{n} \binom{t-1}{2}|V_t| \ge 0$. If at least three players have two or more strategies, then the number of players having one or more positive strategies is at least three. This implies that $|V_3| > 0$, so we have $B_n(k_1, \ldots, k_n) > A_n(k_1, \ldots, k_n)$.

Second alternative proof: Denote $\phi_n := (\prod_{i=1}^{n} q_i)[B_n(k_1, \ldots, k_n) - A_n(k_1, \ldots, k_n)]$. Since $q_i = 1 - p_i$, from (22) and (26), we have

$$\phi_n = \sum_{1 \le i < j \le n} p_i p_j - \prod_{i=1}^{n} (1 - p_i) - \sum_{i=1}^{n} p_i + 1 \tag{33}$$

For $n = 2$, we have $\phi_2 = p_1 p_2 - (1 - p_1)(1 - p_2) - (p_1 + p_2) + 1 = 0$. For $n = 3$, we have $\phi_3 = p_1 p_2 + p_1 p_3 + p_2 p_3 - (1 - p_1)(1 - p_2)(1 - p_3) - (p_1 + p_2 + p_3) + 1 = p_1 p_2 p_3 \ge 0$ and it is positive if $p_i = (k_i - 1)/k_i > 0$ for all $i = 1, 2, 3$, that is, if all three players have two or more strategies. Observe from (33) that

$$\phi_{n+1} = \sum_{1 \le i < j \le n+1} p_i p_j - \prod_{i=1}^{n+1} (1 - p_i) - \sum_{i=1}^{n+1} p_i + 1 = (1 - p_{n+1})\phi_n + p_{n+1} \sum_{1 \le i < j \le n} p_i p_j \tag{34}$$

Since $\phi_3 \ge 0$, it follows from (34) that $\phi_{n+1} \ge 0$ for all $n \ge 2$.

Suppose there are $n + 1$ players where $n \ge 2$. We prove that if at least three out of $n + 1$ players have two or more strategies (that is, $p_i > 0$ for at least three i), then $\phi_{n+1} > 0$. The result is true for $n = 2$. We prove the result by induction on n.

If $p_{n+1} = 0$, then (34) implies $\phi_{n+1} = \phi_n$. Since there are at least three $i \in \{1, \ldots, n + 1\}$ such that $p_i > 0$ and $p_{n+1} = 0$, we must have at least three $i \in \{1, \ldots, n\}$ such that $p_i > 0$. But then by induction hypothesis $\phi_n > 0$ and so $\phi_{n+1} = \phi_n > 0$.

Next suppose $p_{n+1} > 0$. Then there are at least two $i \in \{1, \ldots, n\}$ such that $p_i > 0$. Hence $p_{n+1} \sum_{1 \leq i < j \leq n} p_i p_j > 0$ and by (34) we have $\phi_{n+1} > 0$. ∎

Proof of Theorem 3 The proof proceeds in two parts. In part (A) we show that the storage for the algorithm is bounded above by

$$\beta := \max_{t \in \{0,1,\ldots,n-1\}} [2(|V_t| + |V_{t+1}|) + 2t + 1]. \tag{35}$$

Given part (A), in part (B) we show that either $\beta \leq \beta_0$ or $\beta \leq \beta_1$. The proof of part (A) is constructive[5] and done later. Let us first show the proof of part (B) given part (A).

Proof of part (B) given part (A) Note by (7) and (9) that $|V_t| + |V_{t+1}| = [\Pr(Y = t) + \Pr(Y = t+1)] / \prod_{i=1}^{n} q_i$ where Y follows the Poisson's binomial distribution given in Section 2.2. Darroch (1964) has shown that the mode of a Poisson's binomial distribution distribution differs from its mean by at most 1. Specifically, for a Poisson's binomial variable Y, the mode is either $\lfloor E(Y) \rfloor$ or $\lfloor E(Y) \rfloor + 1$ or both, where $\lfloor E(Y) \rfloor$ is the largest integer not exceeding $E(Y)$. Using Darroch's result, $|V_t| + |V_{t+1}|$ is bounded above by either $2\Pr(Y = \lfloor E(Y) \rfloor) / \prod_{i=1}^{n} q_i$ or by $2\Pr(Y = \lfloor E(Y) \rfloor + 1) / \prod_{i=1}^{n} q_i$. Noting that $E(Y) = \sum_{i=1}^{n} p_i$ and $t \leq n - 1$, it follows from (35) that either $\beta \leq \beta_0$ or $\beta \leq \beta_1$.

Proof of part (A) There are 4 objects at each step of executing the algorithm: (i) vertices, (ii) weights associated with vertices, (iii) edges and (iv) weights associated with edges. Recall that $N_\tau(v)$ = number of vertices, $N_\tau(w_v)$ = number of weights associated with vertices, $N_\tau(e)$ = number of edges and $N_\tau(w_e)$ = number of weights associated with edges to run step τ of the algorithm. The storage requirement at step τ is $SR_\tau = N_\tau(v) + N_\tau(w_v) + N_\tau(e) + N_\tau(w_e)$ (as the step τ will be clear from the context, henceforth we drop the subscript τ).

We begin from the singleton set $V_0 = \{z\}$. Since there is no path terminating at z, path independence vacuously holds for z. In the initial step of the algorithm, construct a graph with the sole vertex z and no edge.

Step 0

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\lvert V_0 \rvert = 1$	0	0	0	1

Step 1

At Step 1, we verify path independence for the vertices of V_1. At step 1, we have two sub-steps for every vertex $s \in V_1$. In sub-step 1(a) we add vertex s, add the only edge that originates from z and terminates at s together its weight. As there is only one path from z to s, path independence trivially holds for any $s \in V_1$. In sub-step 1(b), we delete the edge as well as its weight and put that weight on the vertex s (this weight is the common length of all paths from z to s). So we have

[5] In the end of the paper it is shown how the storage requirement of the algorithm is determined for the game Γ_1 of Example 1.

Step 1: first vertex of V_1

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
(a): $\|V_0\|+1$ (add a vertex s)	0	$0+1=1$ (add the edge from z to s)	$0+1=1$ (add weight of edge)	$\|V_0\|+3$
(b): $\|V_0\|+1$	$0+1=1$ (add weight for s)	$1-1=0$ (delete edge)	$1-1=0$ (delete weight of edge)	$\|V_0\|+2$

Continuing from the last table, for the next vertex of V_1, we have

Step 1: next vertex of V_1

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
(a): $(\|V_0\|+1)+1$ $=\|V_0\|+2$	1	$0+1=1$	$0+1=1$	$\|V_0\|+5$
(b): $\|V_0\|+2$	$(\|V_0\|+1)+1$ $=\|V_0\|+2$	$1-1=0$	$1-1=0$	$\|V_0\|+4$

Following similar steps, for the last vertex in V_1, we have

Step 1: last vertex of V_1

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
(a): $(\|V_0\|+\|V_1\|-1)+1$ $=\|V_0\|+\|V_1\|$	$\|V_1\|-1$	$0+1=1$	$0+1=1$	$\|V_0\|+2\|V_1\|+1$ $<2(\|V_0\|+\|V_1\|)+(2\times 0)+1$
(b): $\|V_0\|+\|V_1\|$	$(\|V_1\|-1)+1$ $=\|V_1\|$	$1-1=0$	$1-1=0$	$\|V_0\|+2\|V_1\|$

Once PI has been established for vertices in V_0 and V_1, to verify PI for vertices in V_t ($t\geq 2$) we only need vertices of V_1 and their associated weights (where the weight for a vertex $s\in V_1$ is the length of the only path from z to s). So the sole vertex of V_0 is deleted in the end of step 1 and we have:

In the end of step 1

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\|$	$\|V_1\|$	0	0	$2\|V_1\|$

At step $t\geq 2$, we have two sub-steps for every vertex $s\in V_t$: (a) we add vertex s, add the t edges that terminate to s from its predecessors together with their weights and (b) after verifying these weights are equal (which requires $t-1$ equations), we delete the edges as well as their weights and put the single weight on the vertex s (this weight is the common length of all paths from z to s). Since any vertex in V_2 has 2 predecessors, when we go to Step 2 from Step 1 to add the first vertex s of V_2, we have

Step 2: first vertex of V_2

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
(a): $\|V_1\|+1$ (add a vertex s)	$\|V_1\|$	$0+2=2$ (add edges terminating at s)	$0+2=2$ (add weights of edges)	$2\|V_1\|+5$
(b): $\|V_1\|+1$	$\|V_1\|+1$ (add weight for s)	$2-2=0$ (delete edges)	$2-2=0$ (delete weights of edges)	$2\|V_1\|+2$

Continuing from the last table, for the next vertex of V_2, we have

Step 2: next vertex of V_2

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
(a): $(\lvert V_1\rvert+1)+1$ $=\lvert V_1\rvert+2$	$\lvert V_1\rvert+1$	$0+2=2$	$0+2=2$	$2\lvert V_1\rvert+7$
(b): $\lvert V_1\rvert+2$	$(\lvert V_1\rvert+1)+1$ $=\lvert V_1\rvert+2$	$2-2=0$	$2-2=0$	$2\lvert V_1\rvert+4$

Following similar steps, for the last vertex in V_2, we have

Step 2: last vertex of V_2

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
(a): $(\lvert V_1\rvert+\lvert V_2\rvert-1)+1$ $=\lvert V_1\rvert+\lvert V_2\rvert$	$\lvert V_1\rvert+\lvert V_2\rvert-1$	$0+2=2$	$0+2=2$	$2(\lvert V_1\rvert+\lvert V_2\rvert)+3=$ $2(\lvert V_1\rvert+\lvert V_2\rvert)+(2\times 1)+1$
(b): $\lvert V_1\rvert+\lvert V_2\rvert$	$(\lvert V_1\rvert+\lvert V_2\rvert-1)+1$ $=\lvert V_1\rvert+\lvert V_2\rvert$	$2-2=0$	$2-2=0$	$2(\lvert V_1\rvert+\lvert V_2\rvert)$

Once PI has been established for vertices in V_1 and V_2, to verify PI for vertices in V_t $(t\geq 3)$ we only need vertices of V_2 and their associated weights (where the weight for a vertex $s\in V_2$ is the common length of any path from z to s). So the vertices of V_1 and their weights are deleted after step 2 and we have:

In the end of step 2

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\lvert V_2\rvert$	$\lvert V_2\rvert$	0	0	$2\lvert V_2\rvert$

By similar reasoning, we have the following in the end of step t

In the end of step t

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\lvert V_t\rvert$	$\lvert V_t\rvert$	0	0	$2\lvert V_t\rvert$

In step $t+1$, we add vertices of V_{t+1}, where for each vertex there are two sub-steps (a),(b) as before. Since $t+1$ edges terminate to any vertex in V_{t+1}, step $t+1$ proceeds as follows:

Step $t+1$

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
(a): $\lvert V_t\rvert+1$	$\lvert V_t\rvert$	$0+(t+1)$ $=t+1$	$0+(t+1)$ $=t+1$	$2\lvert V_t\rvert+1+2(t+1)$
(b): $\lvert V_t\rvert+1$	$\lvert V_t\rvert+1$	$(t+1)$ $-(t+1)=0$	$(t+1)$ $-(t+1)=0$	$2\lvert V_t\rvert+2$
$\vdots$				
(a): $(\lvert V_t\rvert+\lvert V_{t+1}\rvert-1)$ $+1=\lvert V_t\rvert+\lvert V_{t+1}\rvert$	$\lvert V_t\rvert+\lvert V_{t+1}\rvert-1$	$0+(t+1)$ $=t+1$	$0+(t+1)$ $=t+1$	$2(\lvert V_t\rvert+\lvert V_{t+1}\rvert)+2t+1$
(b): $\lvert V_t\rvert+\lvert V_{t+1}\rvert$	$(\lvert V_t\rvert+\lvert V_{t+1}\rvert-1)$ $+1=\lvert V_t\rvert+\lvert V_{t+1}\rvert$	$(t+1)$ $-(t+1)=0$	$(t+1)$ $-(t+1)=0$	$2(\lvert V_t\rvert+\lvert V_{t+1}\rvert)$

Note that the storage requirement (SR) in step 0 is $|V_0| = 1$. The maximum SR in step 1 is lower than $2(|V_0| + |V_1|) + (2 \times 0) + 1$ (see the table corresponding to "Step 1: the last vertex of V_1"). The maximum SR in step 2 is $2(|V_1| + |V_2|) + (2 \times 1) + 1$ (see the table corresponding to "Step 2: the last vertex of V_2"). In general, for $t \geq 1$, the maximum SR in step $t + 1$ is $2(|V_t| + |V_{t+1}|) + 2t + 1$ (see the second last row of the table corresponding to "Step $t + 1$"). This shows that the storage requirement of the algorithm is bounded above by $\beta = \max_{t \in \{0,1,\ldots,n-1\}}[2(|V_t| + |V_{t+1}|) + 2t + 1]$. ■

Acknowledgements

I express my sincere gratitude to the editor and two anonymous reviewers for their helpful comments and suggestions. I am also grateful to the seminar participants of the 22nd International Conference on Game Theory at Stony Brook for their comments on an earlier version of the paper. Research support by the Faculty of Arts, Ryerson University is gratefully acknowledged.

References

Anderson, S.P., Goeree, J.P., Holt, C.A. 2001. Minimum-effort coordination games: Stochastic potential and logit equilibrium. *Games and Economic Behavior*, 34: 177-199

Bramoullé, Y. 2007. Anti-coordination and social interactions. *Games and Economic Behavior*, 58: 30-49

Bramoullé, Y., Kranton, R., D'Amours, M. 2014. Strategic interaction and networks. *American Economic Review*, 104: 898-930

Cheng, D., Liu, T., Zhang, K., Qi, H. 2016. On decomposed subspaces of finite games. *IEEE Transactions on Automatic Control*, 61: 3651-3656

Chien, S., Sinclair, A. 2011. Convergence to approximate Nash equilibria in congestion games. *Games and Economic Behavior*, 71: 315-327

Darroch, J.N. 1964. On the distribution of the number of successes in independent trials. *Annals of Mathematical Statistics*, 35: 1317-1321

Ellingsæter, B., Skjegstad, M., Maseng, T. 2012. A potential game for power and frequency allocation in large-scale wireless networks, arXiv preprint arXiv:1212.0724, pp. 1-10, 2012. Available: http://arxiv.org/abs/1212.0724

Hino, Y. 2011. An improved algorithm for detecting potential games. *International Journal of Game Theory*, 40: 199-205

Hoeffding, W. 1956. On the distribution of the number of successes in independent trials. *Annals of Mathematical Statistics*, 27: 713-721

Marden, J.R., Arslan, G., Shamma, J.S. 2009. Cooperative control and potential games. *IEEE Transactions on Systems, Man and Cybernetics-part B*, 39: 1393-1407

Monderer, D., Shapley, L.S. 1996. Potential games. *Games and Economic Behavior*, 14: 124-143

Neel, J.O., Reed, J.H., Gilles, R.P. 2004. Convergence of cognitive radio networks, in Proceedings of IEEE Wireless Communications Network Conference (WCNC), Atlanta, GA, Mar. 2004, 2250-2255

Pitman, J. 1997. Probabilistic bounds on the coefficients of polynomials with only real zeros. *Journal of Combinatorial Theory, Series A*, 77: 279-303

Roughgarden, T., Tardos, E. 2002. How bad is selfish routing? *Journal of the ACM* 49: 236-259

Sandholm, W.H. 2010. Decompositions and potentials for normal form games. *Games and Economic Behavior*, 70: 446-456

Todd, M.J. 2016. Computation, multiplicity, and comparative statics of Cournot equilibria in integers. *Mathematics of Operations Research*, 41: 1125-1134

Wang, Y.H. 1993. On the number of successes in independent trials. *Statistica Sinica*, 3: 295-312

Diagram 1: directed graph presentation of the game Γ_1 of Example 1

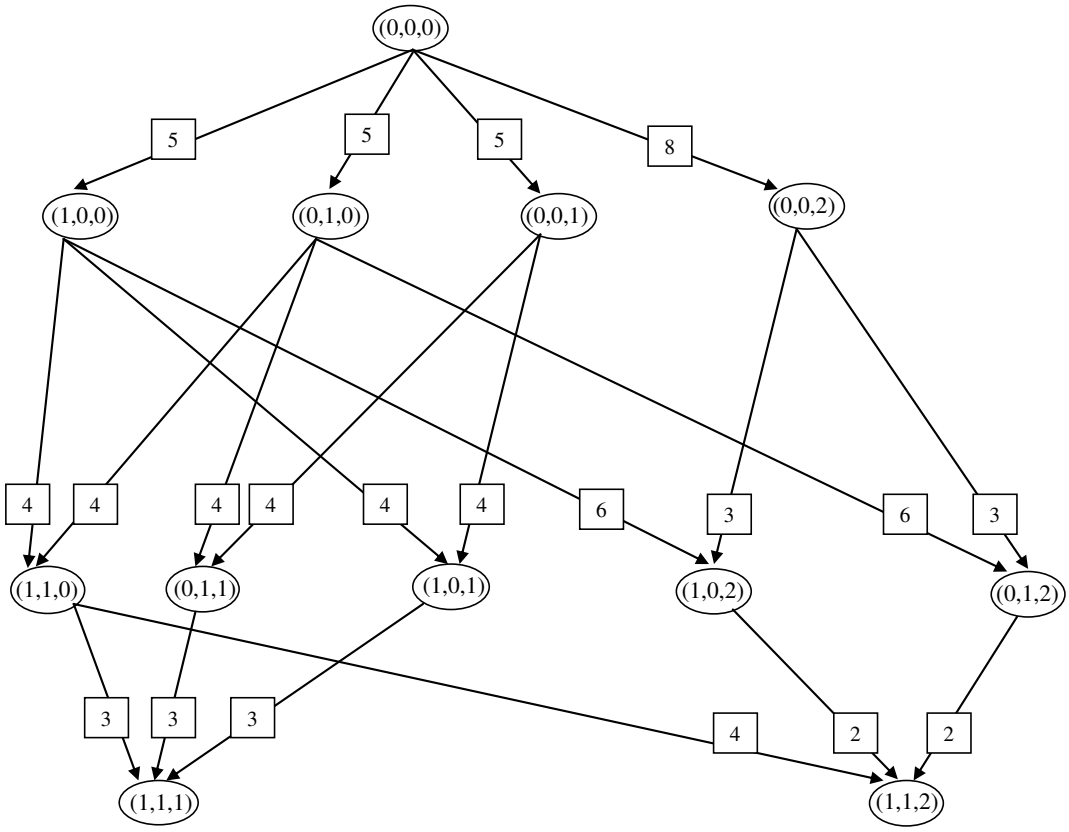
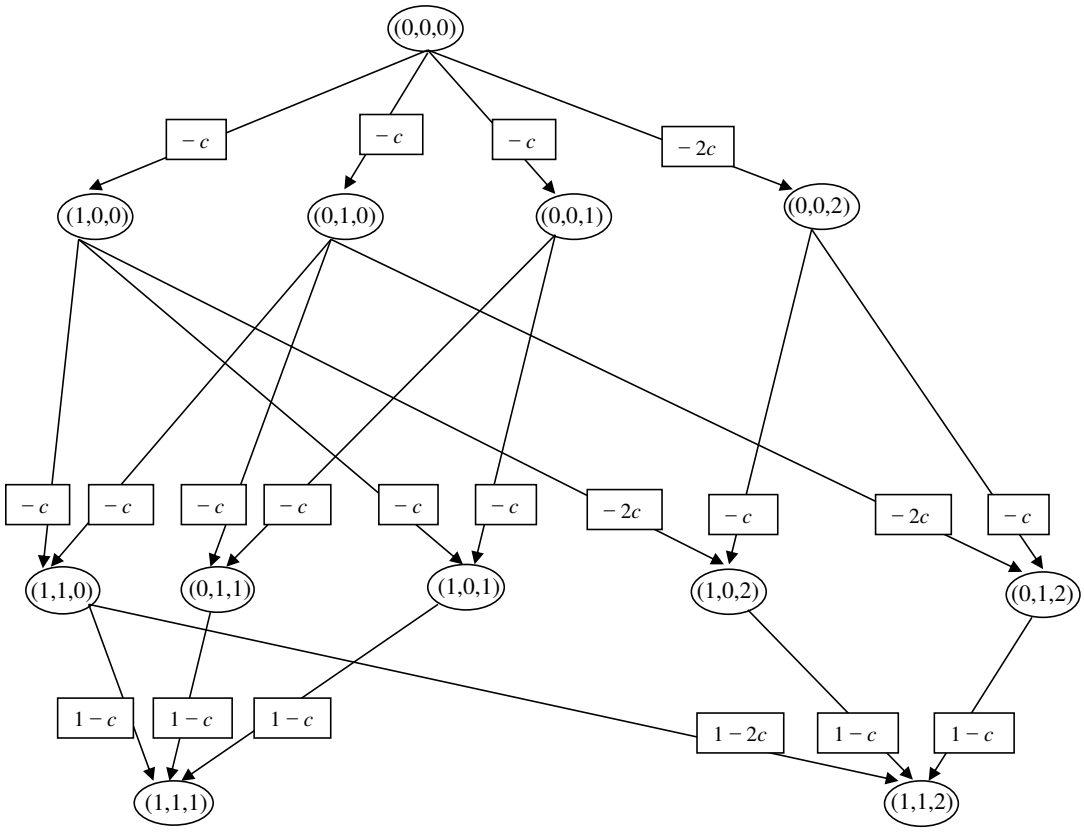


Diagram 2: directed graph presentation of the game Γ_2 of Example 2



Determination of storage requirement for the game Γ_1 (Example 1)

4 types of objects are needed at any step of the algorithm: vertices, weights associated with vertices, edges and weights associated with vertices. Denote by $N(v)$ = number of vertices, $N(w_v)$ = number of weights associated with vertices, $N(e)$ = number of edges, $N(w_e)$ = number of weights associated with edges. The storage requirement at any step of the algorithm is $SR = N(v) + N(w_v) + N(e) + N(w_e)$.

Step 0: The sole vertex $z = (0,0,0)$ of V_0:

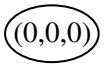


Table 0: Storage requirement in step 0

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_0\| = 1$	0	0	0	$\|V_0\|$

Step 1: First vertex (1,0,0) of V_1

Substep (a): Add vertex (1,0,0), edge between (0,0,0) and (1,0,0) and weight on the edge:

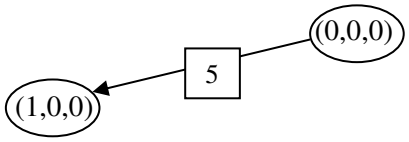


Table 1.1(a): Storage requirement in substep (a)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_0\| + 1$	0	0 + 1= 1	0 + 1= 1	$\|V_0\| + 3$

Substep (b): There is only one path between (0,0,0) and (1,0,0). The length of that path is 5. So path independence (PI) trivially holds for (1,0,0). We delete the edge (after verifying PI, keeping the edge is not necessary). We also delete the weight on the edge and put the weight on the vertex (1,0,0) as below. The weight on (1,0,0) stands for the length of the only path between (0,0,0) and (1,0,0).

Table 1.1(b): Storage requirement in substep (b)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_0\| + 1$	0 + 1= 1	1−1= 0	1−1= 0	$\|V_0\| + 2$

Second vertex (0,1,0) of V_1:

Substep (a): add vertex (0,1,0), edge between (0,0,0) and (0,1,0) and weight on the edge:

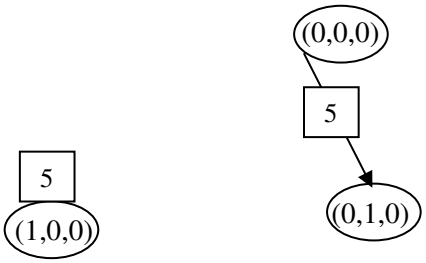


Table 1.2(a): Storage requirement in substep (a)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_0\| + 1+1= \|V_0\| + 2$	1	0 +1 = 1	0 +1 = 1	$\|V_0\| + 5$

Substep (b): As there is only one path between (0,0,0) and (0,1,0), PI holds for (0,1,0). We delete the edge, delete the weight on the edge and put the weight on the (0,1,0) as below.

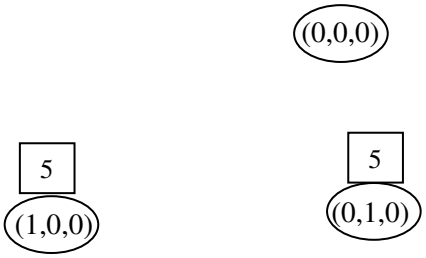


Table 1.2(b): Storage requirement in substep (b)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_0\| + 2$	1 + 1 = 2	1−1= 0	1−1= 0	$\|V_0\| + 4$

Third vertex (0,0,1) of V_1:

Substep (a): add vertex (0,0,1), edge between (0,0,0) and (0,0,1) and weight on the edge:

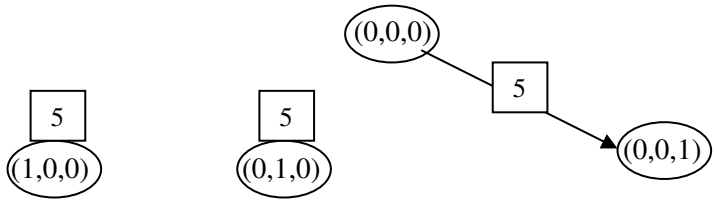


Table 1.3(a): Storage requirement in substep (a)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_0\| +2+1= \|V_0\| + 3$	2	0 +1 = 1	0 +1 = 1	$\|V_0\| + 7$

Substep (b): As there is only one path between (0,0,0) and (0,0,1), PI holds for (0,0,1). We delete the edge, delete the weight on the edge and put the weight on the (0,0,1) as below.

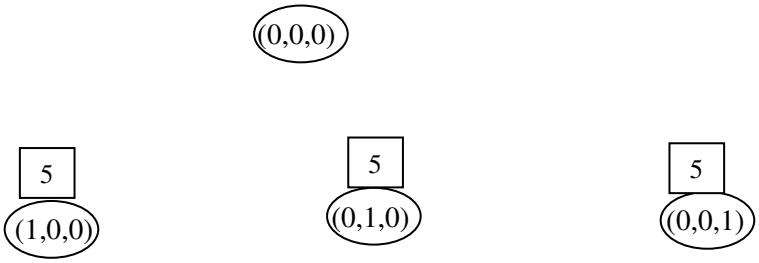


Table 1.3(b): Storage requirement in substep (b)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_0\| + 3$	2 + 1 = 3	1−1= 0	1−1= 0	$\|V_0\| + 6$

Last vertex (0,0,2) of V_1:

Substep (a): add vertex (0,0,2), edge between (0,0,0) and (0,0,2) and weight on the edge

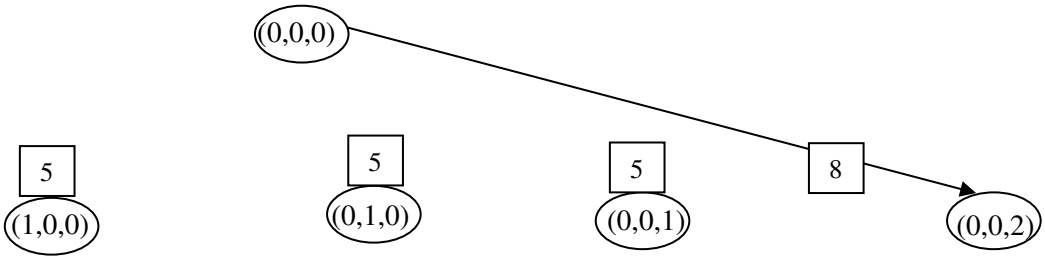


Table 1.4(a): Storage requirement in substep (a)
[this is the maximum *SR* that we can have in Step 1]

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_0\| +3+1= \|V_0\| + 4$ $= \|V_0\| + \|V_1\|$	$3 = \|V_1\|-1$	0 +1 = 1	0 +1 = 1	$\|V_0\| + 9$ $=\|V_0\| + 2\|V_1\| + (2\times0) +1$ $< 2(\|V_0\|+\|V_1\|)+ (2\times0) +1$

Substep (b): As there is only one path between (0,0,0) and (0,0,2), PI holds for (0,0,2). We delete the edge, delete the weight on the edge and put the weight on the (0,0,2) as below.

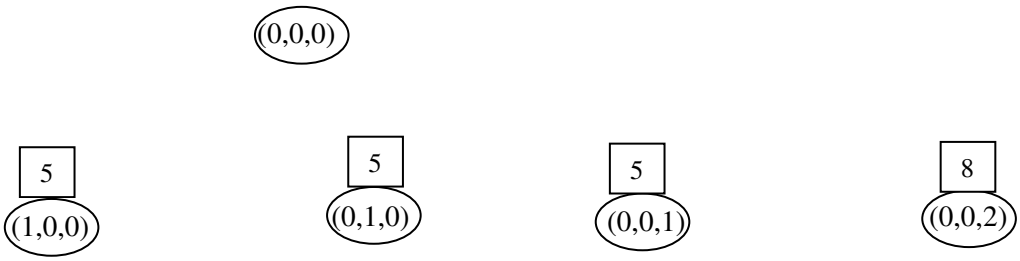


Table 1.4(b): Storage requirement in substep (b)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_0\| + \|V_1\|$	$\|V_1\| -1 + 1 = \|V_1\|$	$1-1= 0$	$1-1= 0$	$\|V_0\| + 2\|V_1\|$

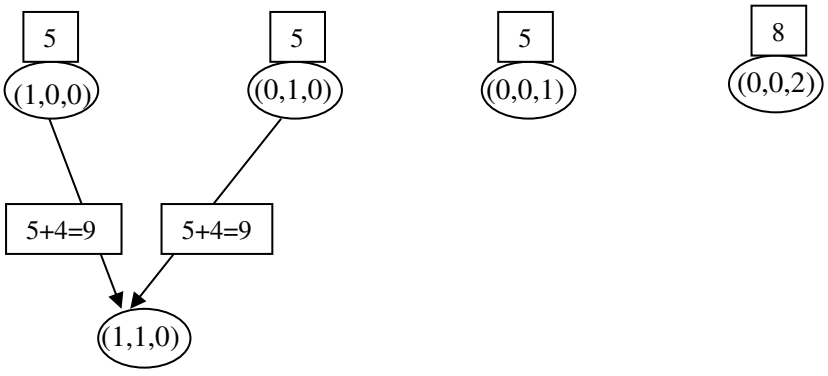
End of step1: As PI has been established for all vertices in V_1, we can delete vertex (0,0,0) and keep only the vertices in V_1 with their associated weights.

Table 1.4(c): Storage requirement in the end of step 1

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_0\|+ \|V_1\|-\|V_0\| =\|V_1\|$	$\|V_1\|$	0	0	$2\|V_1\|$

Step 2: First vertex (1,1,0) of V_2:

Substep (a): add vertex (1,1,0), edges terminating at (1,1,0) and weights on the edges



Note that there are two edges terminating at (1,1,0): from (1,0,0) and from (0,1,0). PI has been already established for (1,0,0) and (0,1,0). The common length of any path from z = (0,0,0) to (1,0,0) is 5. As the length of the path between (1,0,0) and (1,1,0) is 4, the length of any path from z to (1,1,0) that goes through (1,0,0) is 5 + 4 = 9. This is the weight we put on the edge between (1,0,0) and (1,1,0). By the same reasoning we put the weight 5 + 4 = 9 on the edge between (0,1,0) and (1,1,0).

Table 2.1(a): Storage requirement in substep (a)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\| + 1$	$\|V_1\|$	0 +2 = 2	0 +2 = 2	$2\|V_1\| +5$

Substep (b): As both edges terminating at (1,1,0) have the same associated weight 9, PI holds for (1,1,0) and the common length of any path between z and (1,1,0) is 9. We delete the two edges, delete the two weights on the edges and put the weight 9 on the vertex (1,1,0) as below.

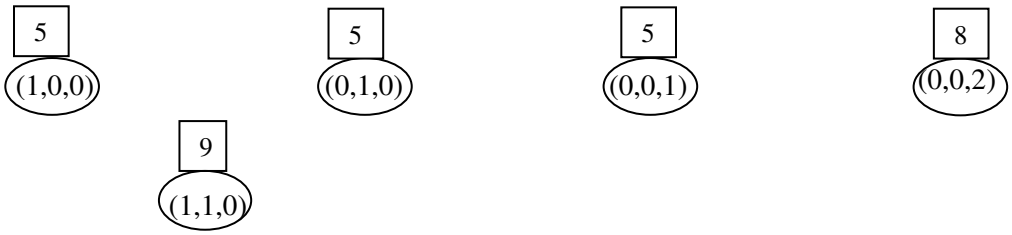
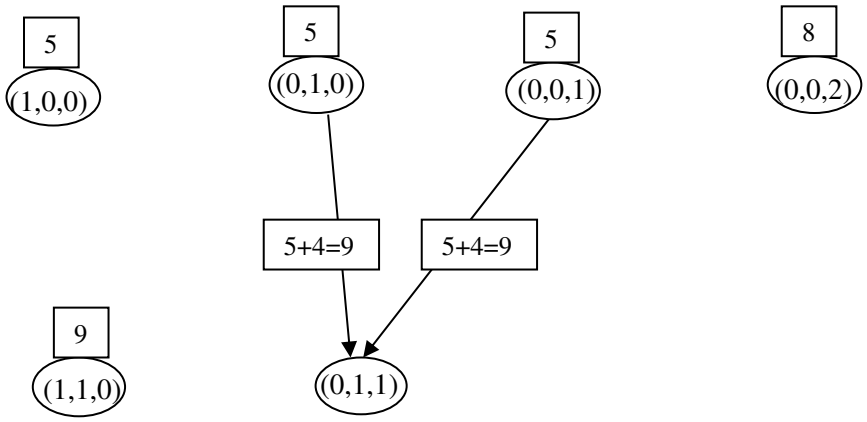


Table 2.1(b): Storage requirement in substep (b)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\| + 1$	$\|V_1\| + 1$	2 −2 = 0	2 −2 = 0	$2\|V_1\| +2$

Second vertex (0,1,1) of V_2:

Substep (a): add vertex (0,1,1), edges terminating at (0,1,1) and weights on the edges



There are two edges terminating at (0,1,1): from (0,1,0) and from (0,0,1). PI has been already established for (0,1,0) and (0,0,1). The common length of any path from z to (0,1,0) is 5. As the length of the path between (0,1,0) and (0,1,1) is 4, the length of any path from z to (0,1,1) that goes through (0,1,0) is 5 + 4 = 9. This is the weight we put on the edge between (0,1,0) and (0,1,1). By the same reasoning we put the weight 5 + 4 = 9 on the edge between (0,0,1) and (0,1,1).

Table 2.2(a): Storage requirement in substep (a)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\| + 1 + 1 = \|V_1\| + 2$	$\|V_1\| + 1$	$0 + 2 = 2$	$0 + 2 = 2$	$2\|V_1\| + 7$

Substep (b): As both edges terminating at (0,1,1) have the same associated weight 9, PI holds for (0,1,1) and the common length of any path between z and (1,1,0) is 9. We delete the two edges, delete the two weights on the edges and put the weight 9 on the vertex (0,1,1) as below.

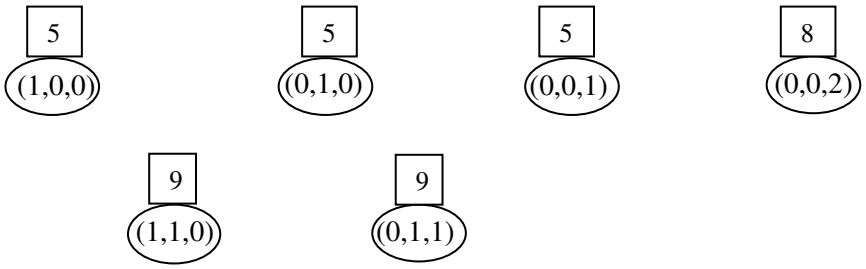
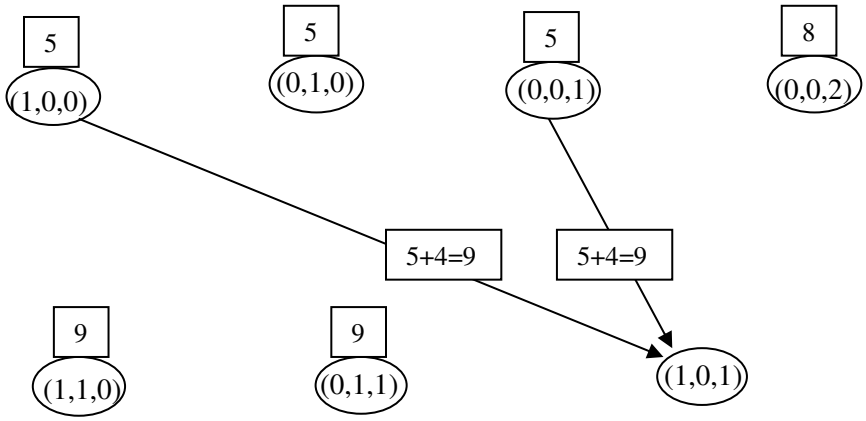


Table 2.2(b): Storage requirement in substep (b)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\| + 2$	$\|V_1\| + 1 + 1 = \|V_1\| + 2$	$2 - 2 = 0$	$2 - 2 = 0$	$2\|V_1\| + 4$

Third vertex (1,0,1) of V_2:

Substep (a): add vertex (1,0,1), edges terminating at (1,0,1) and weights on the edges



There are two edges terminating at (1,0,1): from (1,0,0) and from (0,0,1). By the same reasoning as the previous case, the length of any path from z to (1,0,1) that goes through (1,0,0) is 5 + 4 = 9. The length of any path from z to (1,0,1) that goes through (0,0,1) is also 5 + 4 = 9. These are weights we put on the corresponding edges.

Table 2.3(a): Storage requirement in substep (a)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\| + 2 + 1 = \|V_1\| + 3$	$\|V_1\| + 2$	$0 + 2 = 2$	$0 + 2 = 2$	$2\|V_1\| + 9$

Substep (b): As both edges terminating at (1,0,1) have the same associated weight 9, PI holds for (1,0,1) and the common length of any path between z and (1,0,1) is 9. We delete the two edges, delete the two weights on the edges and put the weight 9 on the vertex (1,0,1) as below.

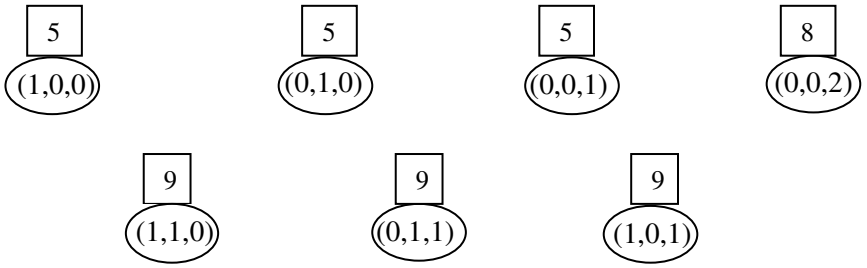
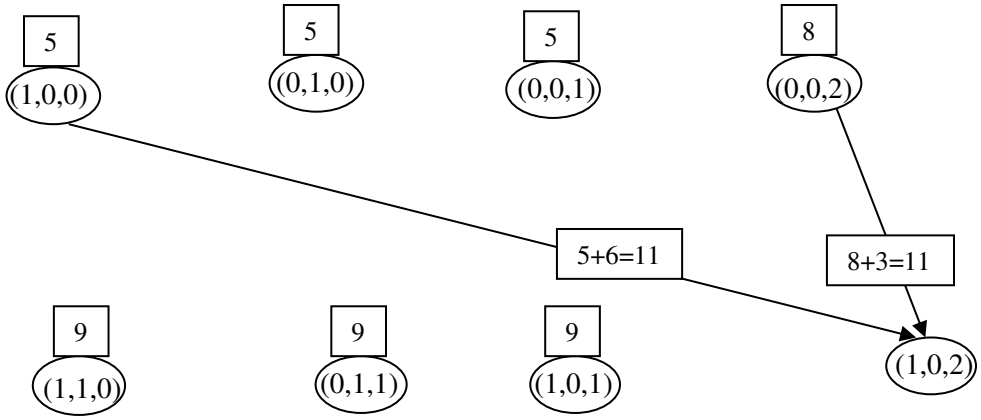


Table 2.3(b): Storage requirement in substep (b)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\| + 3$	$\|V_1\| + 2 + 1 = \|V_1\| + 3$	$2 - 2 = 0$	$2 - 2 = 0$	$2\|V_1\| + 6$

Fourth vertex (1,0,2) of V_2:

Substep (a): add vertex (1,0,2), edges terminating at (1,0,2) and weights on the edges



There are two edges terminating at (1,0,2): from (1,0,0) and from (0,0,2). PI has been already established for (1,0,0) and (0,0,2). The common length of any path from z to (1,0,0) is 5. As the length of the path between (1,0,0) and (1,0,2) is 6, the length of any path from z to (1,0,2) that goes through (1,0,0) is 5 + 6 = 11. This is the weight we put on the edge between (1,0,0) and (1,0,2).

The common length of any path from z to (0,0,2) is 8. As the length of the path between (0,0,2) and (1,0,2) is 3, the length of any path from z to (1,0,2) that goes through (0,0,2) is 8 + 3 = 11. This is the weight we put on the edge between (0,0,2) and (1,0,2).

Table 2.4(a): Storage requirement in substep (a)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\| + 3 + 1 = \|V_1\| + 4$	$\|V_1\| + 3$	$0 + 2 = 2$	$0 + 2 = 2$	$2\|V_1\| + 11$

Substep (b): As both edges terminating at (1,0,2) have the same associated weight 11, PI holds for (1,0,2) and the common length of any path between z and (1,0,1) is 11. We delete the two edges, delete the two weights on the edges and put the weight 11 on the vertex (1,0,2) as below.

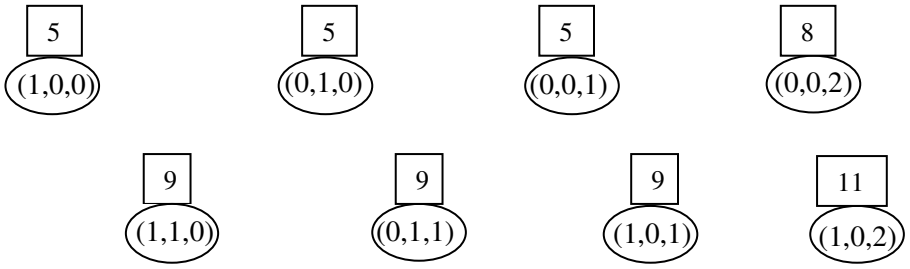
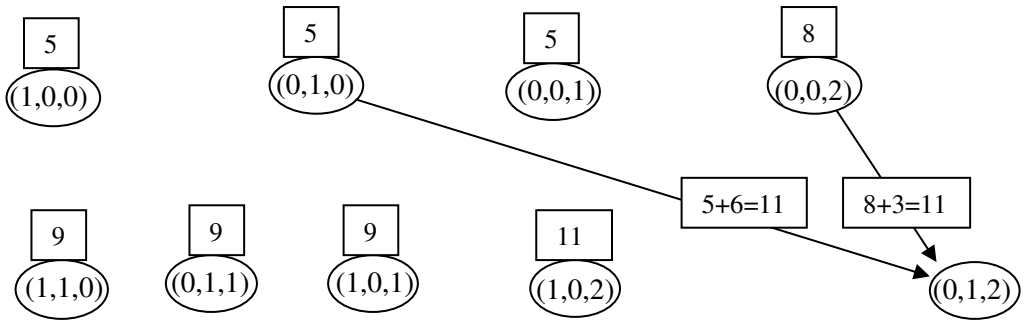


Table 2.4(b): Storage requirement in substep (b)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\| + 4$	$\|V_1\| + 3 + 1 = \|V_1\| + 4$	$2 - 2 = 0$	$2 - 2 = 0$	$2\|V_1\| + 8$

Last vertex (0,1,2) of V_2:

Substep (a): add vertex (0,1,2), edges terminating at (0,1,2) and weights on the edges



There are two edges terminating at (0,1,2): from (0,1,0) and from (0,0,2). PI has been already established for (0,1,0) and (0,0,2). The common length of any path from z to (0,1,0) is 5. As the length of the path between (0,1,0) and (0,1,2) is 6, the length of any path from z to (0,1,2) that goes through (0,1,0) is 5 + 6 = 11. This is the weight we put on the edge between (0,1,0) and (0,1,2).

The common length of any path from z to (0,0,2) is 8. As the length of the path between (0,0,2) and (0,1,2) is 3, the length of any path from z to (0,1,2) that goes through (0,0,2) is 8 + 3 = 11. This is the weight we put on the edge between (0,0,2) and (0,1,2).

Table 2.5(a): Storage requirement in substep (a)
[this is the maximum *SR* that we can have in Step 2]

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\| + 4 + 1 = \|V_1\| + 5$ $= \|V_1\| + \|V_2\|$	$\|V_1\| + 4$ $= \|V_1\| + \|V_2\| - 1$	$0 + 2 = 2$	$0 + 2 = 2$	$2\|V_1\| + 13$ $= 2(\|V_1\| + \|V_2\|) + 3$ $= 2(\|V_1\| + \|V_2\|) + (2\times 1) + 1$

Substep (b): As both edges terminating at (0,1,2) have the same associated weight 11, PI holds for (0,1,2) and the common length of any path between z and (0,1,2) is 11. We delete the two edges, delete the two weights on the edges and put the weight 11 on the vertex (0,1,2) as below.

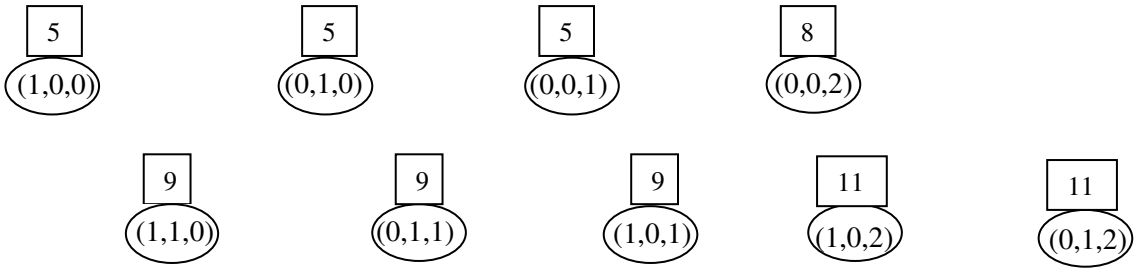


Table 2.5(b): Storage requirement in substep (b)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\| + \|V_2\|$	$\|V_1\| + \|V_2\| - 1 + 1$ $= \|V_1\| + \|V_2\|$	$2 - 2 = 0$	$2 - 2 = 0$	$2(\|V_1\| + V_2\|)$

End of step 2: As PI has been established for all vertices in V_2, we can delete the vertices in V_1 and keep only the vertices in V_2 with their associated weights.

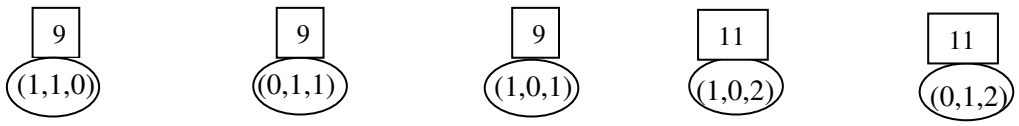
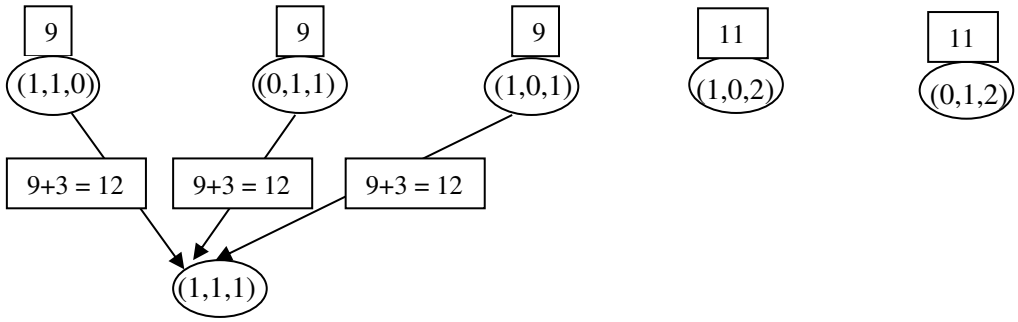


Table 2.5(c): Storage requirement in the end of step 2

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_1\| + \|V_2\| - \|V_1\|$ $= \|V_2\|$	$\|V_1\| + \|V_2\| - \|V_1\|$ $= \|V_2\|$	0	0	$2\|V_2\|$

Step 3: First vertex (1,1,1) of V_3:

Substep (a): add vertex (1,1,1), edges terminating at (1,1,1) and weights on the edges



There are three edges terminating at (1,1,1): from (1,1,0), from (0,1,1) and from (1,0,1). PI has been already established for (1,1,0), (0,1,1), (1,0,1). The common length of any path from z to (1,1,0) is 9. As the length of the path between (1,1,0) and (1,1,1) is 3, the length of any path from z to (1,1,1) that goes through (1,1,0) is 9 + 3 = 12. This is the weight we put on the edge between (1,1,0) and (1,1,1).

The common length of any path from z to (0,1,1) is 9. As the length of the path between (0,1,1) and (1,1,1) is 3, the length of any path from z to (1,1,1) that goes through (0,1,1) is 9 + 3 = 12. This is the weight we put on the edge between (0,1,1) and (1,1,1). By the same reasoning, the weight we put on the edge between (1,0,1) and (1,1,1) is 9 + 3 = 12.

Table 3.1(a): Storage requirement in substep (a)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_2\| + 1$	$\|V_2\|$	0 + 3= 3	0 + 3 = 3	$2\|V_2\| + 7$

Substep (b): As all three edges terminating at (1,1,1) have the same associated weight 12, PI holds for (1,1,1) and the common length of any path between z and (1,1,1) is 12. We delete the three edges, delete the three weights on the edges and put the weight 12 on the vertex (1,1,1) as below.

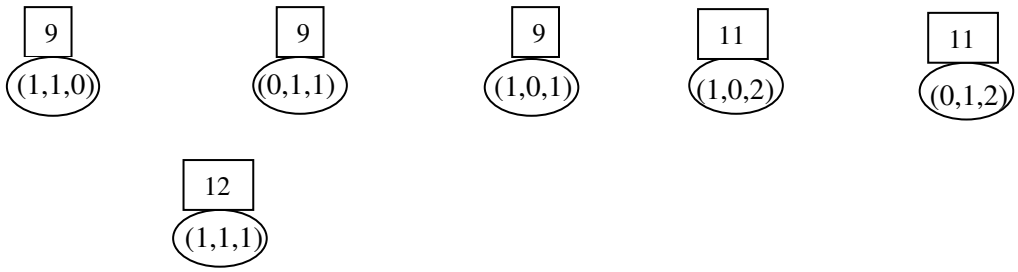
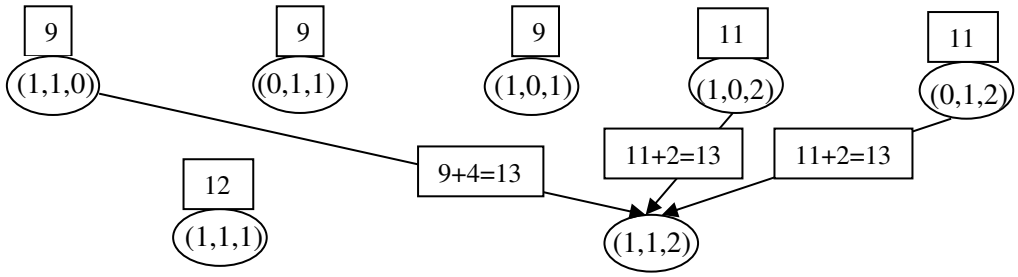


Table 3.1(b): Storage requirement in substep (b)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_2\| + 1$	$\|V_2\| + 1$	3 − 3= 0	3 − 3= 0	$2\|V_2\| + 2$

Last vertex (1,1,2) of V_2:

Substep (a): add vertex (1,1,2), edges terminating at (1,1,2) and weights on the edges



There are three edges terminating at (1,1,2): from (1,1,0), from (1,0,2) and from (0,1,2). PI has been already established for (1,1,0), (1,0,2), (0,1,2). The common length of any path from z to (1,1,0) is 9. As the length of the path between (1,1,0) and (1,1,2) is 4, the length of any path from z to (1,1,2) that goes through (1,1,0) is 9 + 4 = 13. This is the weight we put on the edge between (1,1,0) and (1,1,2).

The common length of any path from z to (1,0,2) is 11. As the length of the path between (1,0,2) and (1,1,2) is 2, the length of any path from z to (1,1,2) that goes through (1,0,2) is 11 + 2 = 13. This is the weight we put on the edge between (1,0,2) and (1,1,2). By the same reasoning, the weight we put on the edge between (0,1,2) and (1,1,2) is 11 + 2 = 13.

Table 3.2(a): Storage requirement in substep (a)
[this is the maximum *SR* that we can have in Step 3]

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_2\| + 1+1=\|V_2\| + 2 = \|V_2\| + \|V_3\|$	$\|V_2\|+1 = \|V_2\| + \|V_3\|-1$	$0 + 3= 3$	$0 + 3 = 3$	$2\|V_2\| + 9$ $= 2(\|V_2\|+\|V_3\|) + 5$ $= 2(\|V_2\|+\|V_3\|) + (2\times2) + 1$

Substep (b): As all three edges terminating at (1,1,2) have the same associated weight 13, PI holds for (1,1,2) and the common length of any path between z and (1,1,2) is 13. We delete the three edges, delete the three weights on the edges and put the weight 13 on the vertex (1,1,2) as below.

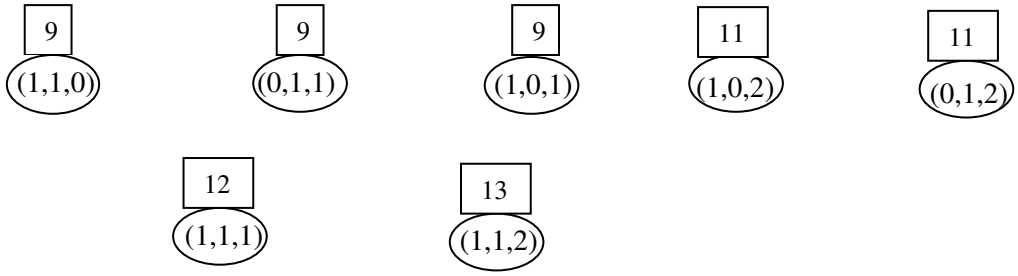


Table 3.2(b): Storage requirement in substep (b)

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_2\| + \|V_3\|$	$\|V_2\| + \|V_3\| - 1 + 1 = \|V_2\| + \|V_3\|$	$3 - 3 = 0$	$3 - 3 = 0$	$2(\|V_2\| + \|V_3\|)$

End of step 3: As PI has been established for all vertices in V_3, we can delete the vertices in V_2 and keep only the vertices in V_3 with their associated weights.

Table 3.2(c): Storage requirement in the end of step 3

$N(v)$	$N(w_v)$	$N(e)$	$N(w_e)$	SR
$\|V_2\| + \|V_3\| - \|V_2\| = \|V_3\|$	$\|V_2\| + \|V_3\| - \|V_2\| = \|V_3\|$	0	0	$2\|V_3\|$

This is the end of the algorithm for the game Γ_1 (Example 1).

Maximum storage requirement:

In step 0, the storage requirement is $|V_0| = 1$.

As shown in Table 1.4(a), in step 1, the maximum storage required is $|V_0| + 2|V_1| + (2\times0) + 1 < 2(|V_0| + |V_1|) + (2\times0) + 1$

As shown in Table 2.5(a), in step 2, the maximum storage required is $2(|V_1| + |V_2|) + (2\times1) + 1$

As shown in Table 3.2(a), in step 3, the maximum storage required is $2(|V_2| + |V_3|) + (2\times2) + 1$

So the storage requirement for the algorithm is bounded above by $\max_{\{t=0,1,2\}}[2(|V_t| + |V_{t+1}|) + 2t + 1]$.