Electronic Colloquium on Computational Complexity, Report No. 121 (2014)

Graph Structure and Parity Games

Sebastian Müller

National Institute of Informatics, Tokyo**

Abstract. We investigate the possible structural changes one can perform on a game graph without destroying the winning regions of the players playing a parity game on it. More precisely, given a game graph (G, p) for which we can efficiently compute winning regions, can we remove or add a vertex or edge or change a single priority and maintain at least part of the winning region? Also, what about restricted classes of graphs, such as planar, k-connected and the like?
Unfortunately we find that for any such class there are simple examples where structural alterations make computing winning regions infeasible (assuming that $\mathsf{ParityGames} \notin \mathbf{P}$). That means the class of efficiently solvable parity games, even on restricted classes of graphs, is not closed under even the simplest structural transformations.

1 Introduction

Parity Games have been thoroughly investigated for several decades, yet the main question, whether computing winning regions is feasible (i.e. doable in polynomial time), remains almost untouched. Interest in these games, on the other hand, is steadily increasing as many applications are known. The most prominent probably being that the model checking problem for the modal μ-calculus and the word and emptiness problem for alternating tree automata can be reduced to deciding winners in parity games (see [4]). For a thorough introduction see [2].

Since all but the very simplest of graph classes have not been shown to be feasible, we are interested in general closure properties of feasible classes, i.e. what transformations can we perform while maintaining feasibility.

2 Preliminaries

A *Parity Game* is a two player game, played on a directed (finite) graph $G = (V_1 \cup V_2, E)$ equipped with a priority function $p : V_1 \cup V_2 \longrightarrow \mathbb{N}$. We call the pair (G, p) a *game graph* and will usually omit mentioning p for the sake of readability. The disjoint vertex sets V_1 and V_2 belong to Player 1 and Player 2, respectively. At the start of the game a token is placed on a specified vertex v_s and then the Players take terms to move the token along the edges of G. At any step of the game, the player on whose vertex the token resides, chooses an adjacent vertex and moves the token there. A *play* is an infinite path on G played according to Player 1's and Player 2's choices. The following conventions can be made on the game graph

- The game graph contains no loops or sinks (vertices with out degree 0).

** email: muller@karlin.mff.cuni.cz

ISSN 1433-8092

– The vertices belonging to Player 1 have odd, the vertices belonging to Player 2 even priorities.

A play is *winning* for Player 1, if the vertex with the highest priority that appears an infinite number of times is odd. Otherwise it is winning for Player 2. A *strategy* for Player i is a function σ_i mapping initial segments of plays which end in V_i, to V_{3-i}. A strategy σ_i for Player i is *winning* iff every play, where Player i moves according to σ_i, is winning for him.

Fact 1 *Parity Games are determined. That is, for any given game graph* (G, p) *and starting vertex* v_s*, either Player 1 or Player 2 has a winning strategy.*

A strategy σ_i is called *memoryless* (or positional) if its function value only depends on the last vertex of the play, i.e. if there exists a function $\sigma'_i : V_i \longrightarrow V_{3-i}$ such that

$$\sigma_i(v_0, v_1, \ldots, v_k) = \sigma'_i(v_k),$$

for all $v_k \in V_i$.

Fact 2 *Parity Games are positionally determined. That is, for any given game graph* (G, p) *and starting vertex* v_s*, either Player 1 or Player 2 has a memoryless winning strategy.*

Thus, we can divide the game graph into a region, from which Player 1 has a winning strategy and one, from which Player 2 has one. We call these regions *winning regions* and denote them by $Win_i(G, p)$. We denote the algorithmic problem to decide if $v_s \in Win_1(G, p)$, given a game graph (G, p) and a starting vertex v_s as input, as $\mathsf{ParityGames}$.

Fact 3 $\mathsf{ParityGames} \in \mathbf{NP} \cap \mathbf{coNP}$.

In [3], Jurdzinski improves this to $\mathsf{ParityGames} \in \mathbf{UP} \cap \mathbf{coUP}$. However, since then little progress was made.

3 Changing the Structure of the Underlying Graph

In this paper we are interested in the impact of structural changes to game graphs on the feasibility of determining winning regions in Parity Games. That is, what changes in the structure of a game graph is possible without altering the complexity of determining the winning regions of the associated parity game?

3.1 Removing a Vertex

Assume we have a game graph (G, p) and wish to remove a single vertex. How will the complexity change if we do so?

Observation 1 *Given a game graph* (G, p)*, there is a game graph* (G', p')*, such that there is a trivial winning strategy in the game associated with* (G', p') *and* G *is an induced subgraph of* G' *resulting from the deletion of two vertices. Moreover the deletion of a single vertex leaves a graph which has the same winning strategies on* (G, p) *as in the game associated with* (G, p)*.*

Proof. We will construct (G', p') as follows. Let v be a new vertex of maximum priority and v' a new vertex of maximal priority $+1$. Without loss of generality, we assume that v is a 1-vertex. We add edges (x, v) for every 2-vertex x, an edge (v, v') and an edge (v', y) for any 1-vertex y (by construction v' is a 2-vertex).

Now, the trivial strategy for Player 2 is to play to v. This is a winning strategy for him. However, deletion of v yields a graph where the strategy has to be computed anew and is equivalent to the strategies on the game over (G, p).

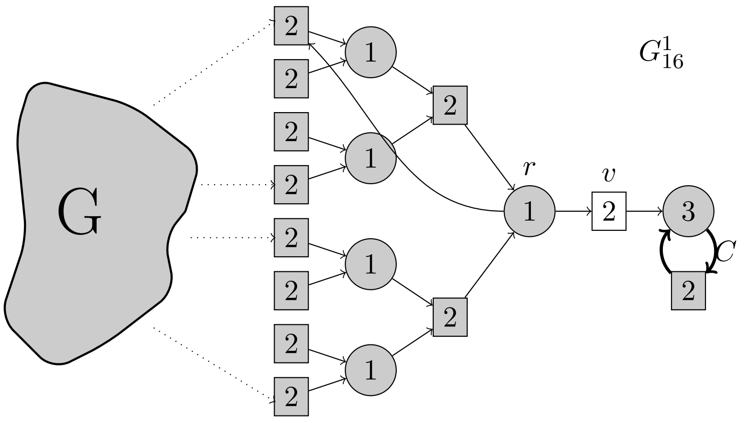
The previous observation says that deletion of vertices can totally destroy a given winning strategy, and thus, the class of Parity Games with a polynomial time strategy are not closed under such structural changes (unless $\mathsf{ParityGames} \in \mathbf{P}$).

However, the deleted vertex has very high in-degree. The next observation shows that also removal of a single constant degree vertex can render a global strategy obsolete.

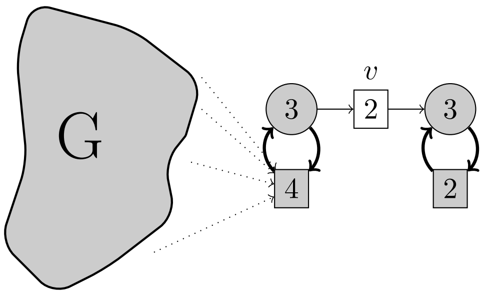
Observation 2 *Given a game graph (G, p), there exists a game graph (G', p') with $|G'| = 3 \cdot |G| + 2$ such that there exists a trivial global winning strategy on (G', p') and the deletion of a single vertex v of in-degree 1 and out-degree 1 leaves a graph for which winning regions can be computed in* **PTIME** *if and only if the same holds for the game associated with (G, p).*

Proof. Given a graph G containing n_1 1-vertices, G' will consist of G together with the following gadget graph $G^1_{n_1}$. $G^1_{n_1}$ consists of an inverse binary tree of even height with $\frac{n_1}{2}$ leafs and a root r (by inverted we mean that an edge (x, y) in the tree is contained in $G^1_{n_1}$ in its inverted direction (y, x)). From r there leads an additional edge to one of the leafs and one edge to a new vertex v. The vertex v has only one outgoing edge to a new cycle C of length 2. The priorities are minimal such that Player 1 wins on the cycle C and 1 and 2 on the inverted tree (i.e. the cycle through r wins for Player 2). We add an edge from at most two 1-vertices in G to any leaf in $G^1_{n_1}$ such that each 1 vertex in G now has an edge to a leaf in $G^1_{n_1}$.

The following picture shows an arbitrary graph G with 16 1-nodes and the corresponding G^1_{16}:



Observe that the graph G' retains many properties of G, as $G^1_{n_1}$ is planar, has constant degree and contains only two cycles. The size is linear in the size of G and we have only added paths of logarithmic length. If one is not interested in the degree of the vertices, one can also collapse the whole tree into a cycle of length 2 and retain the other properties as depicted in the following picture (we will call this gadget $\mathbb{G}_0$):



Obviously, a trivial strategy for Player 1 is to always play into $G^1_{n_1}$ and then go to C. Removal of v, however, makes this strategy obsolete and as 1 always loses when moving to $G^1_{n_1} - \{v\}$, computing winning regions in $G' - \{v\}$ is equivalent to computing them in G.

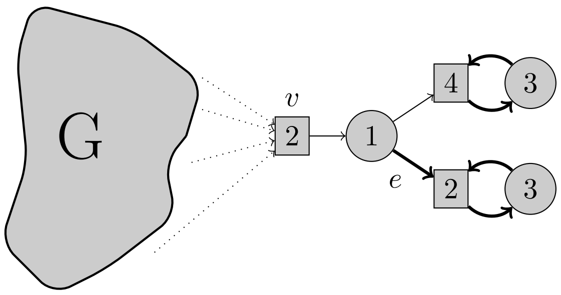
This shows that, unless $\mathsf{ParityGames} \in \mathbf{P}$, the class of Parity Games solvable in $\mathbf{P}$ is not closed under the deletion of vertices, even under rather strong constraints on that vertex and the underlying graph.

3.2 Removing an Edge

Another natural alteration of the structure of a graph is the removal of an edge. It is a similar observation as in the removal of a vertex that this is not possible without having to recompute the Winning Regions.

Observation 3 *For every graph G, there exists a graph G' such that computation of winning regions in G' is trivial and if a single edge e is removed, computation of winning regions in $G' - \{e\}$ is equivalent to computation of winning regions in G.*

Proof. Given G, we let G' be the following graph:

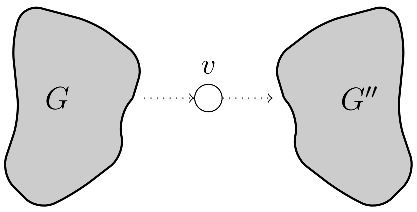


The dotted lines represent an edge from every 0-vertex to v. The trivial strategy is to move to v and then use e to go to the winning circle. Removal of e makes this strategy invalid and we have to compute in G from scratch. As before we can use an inverted tree to get the same result with a linear blowup for graphs with constant degree.

3.3 Adding a Vertex or an Edge

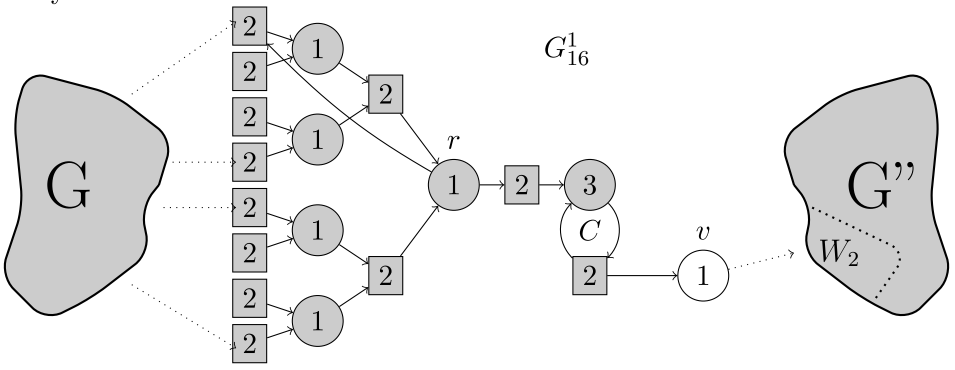
Another natural structural modification of graphs is the addition of vertices. We will see that the addition of only a single vertex under very strong assumptions suffices to make a feasible game as infeasible as possible.

The case we will consider is the easiest perceivable: adding a vertex v with edges coming from G' and edges going to G'', where there is no path between them apart from the ones going through v. The following picture depicts the situation:



Observation 4 *Let $(G, p), (G'', p'')$ be a pair of (not connected) game graphs. Then there is a game graph (G', p') containing (G, p), such that there is a trivial winning strategy in the game over (G', p'), but adding a single vertex v as depicted above makes it at least as hard to compute winning strategies for the new graph $G' \cup \{v\} \cup G''$ as it was in (G, p).*

Proof. Wlog we may assume that Player 2 has a nonempty winning region in G'' and G contain n_1 1-vertices. We let $G' := G \cup G^1_{n_1}$ as above. The new vertex v will have only edges into the winning region of Player 2 in G'' and one edge coming from the 2-vertex in the cycle C:



As before, adding v therefore destroys Player 1's strategy in G' to always play into $G^1_{n_1}$ as it makes any such play an immediate loss for 1.

By adding an edge e instead of a vertex v this also immediately yields the same observation for the case of adding edges:

Observation 5 *Let G, G'' be a pair of (not connected) game graphs. Then there is a directed graph G' containing G, such that there is a trivial winning strategy in G', but adding a single edge e from G' to G'' makes it at least as hard to compute winning strategies for the new graph $G' \cup \{e\} \cup G''$ as it was in G.*

Of course the statement also holds true in the more more general setting of adding vertices or edges into graphs that do not have two disjoint components. We will construct a general gadget for that situation later.

3.4 Changing a Priority

Using the same gadget $G^i_{n_i}$ as above we can directly see that changing a single priority can have a similar effect.

Observation 6 *Given a game graph G, there exists a game graph G' with $|G'| = 3 \cdot |G| + 2$ such that there exists a trivial global winning strategy on G' and changing a single priority leaves a graph for which winning regions can be computed in* **PTIME** *if and only if the same holds for the game over G.*

Proof. Given a graph G, we add the graph $G^1_{n_1}$ as before, getting a graph G' with a trivial strategy for Player 1. We can change a single priority in the cycle C to make it a losing cycle for Player 1 and therefore render the whole gadget useless for him. So his strategy on $G' \upharpoonright G$ is as complicated to compute as it was in G.

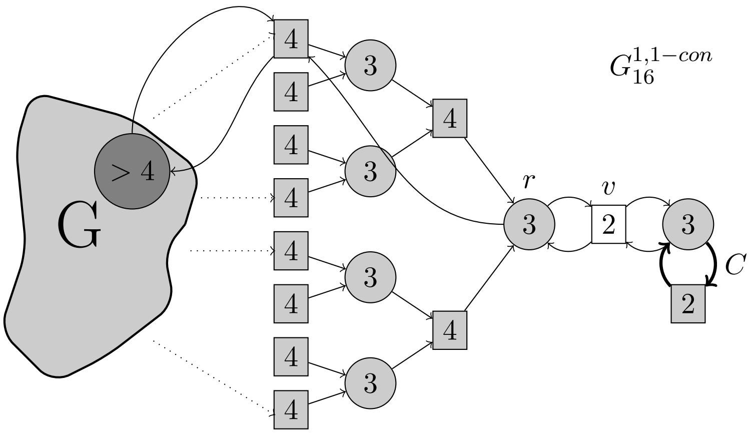
4 Restricted Graph Classes

As we have seen in the previous section, in general it is not possible to alter the structure of the game graph even locally, without possibly losing all of the feasibility of the associated parity game. Moreover, we have seen that the gadgets presented above work for some restricted classes, like graphs of bounded degree or tree width. But what happens for other, natural classes of graphs?

4.1 Graphs with high Connectivity

An important point in constructing the gadgets was their low connectivity. So what can we say about k-connected graphs? Or even graphs whose connectivity depends on the number of vertices, like in the case of the random graph?

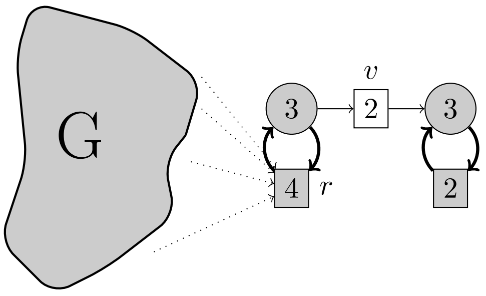
We can modify $G^i_{n_i}$ slightly to a graph we will call $G^{i,1-con}_{n_i}$, so that the whole digraph $G' = G \cup G^{i,1-con}_{n_i}$ retains the property that between any two vertices a path exists (assuming that G had this property). The following picture shows an arbitrary graph G with 16 1-nodes and the corresponding $G^{1,1-con}_{16}$:



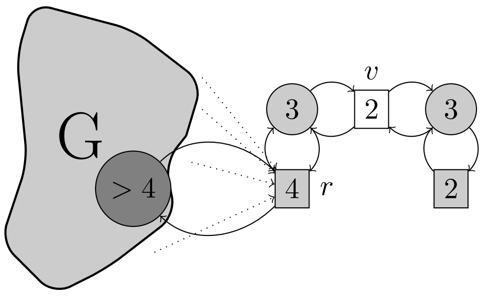
The only assumption we have to make is that G contains one vertex with priority at least 5 for $i = 1$ or 4 for $i = 2$.

Can we show that we can get a similar result as above by removing k vertices or k edges for k connected graphs? If so, is it maybe save to remove $k - 1$ many? To make the following more readable, **we will consider only gadgets built to change the strategy for Player 1. That is, all gadgets we look at will be connected to 1-vertices from the original graph.** The case for Player 2 is verbatim.

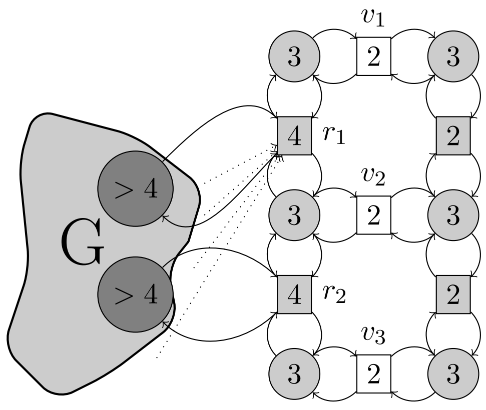
To investigate this we start with the simple gadget $\mathbb{G}_0$ depicted earlier:



Of course we can turn this easily into a gadget $\mathbb{G}_1$ for the 1-connected case:



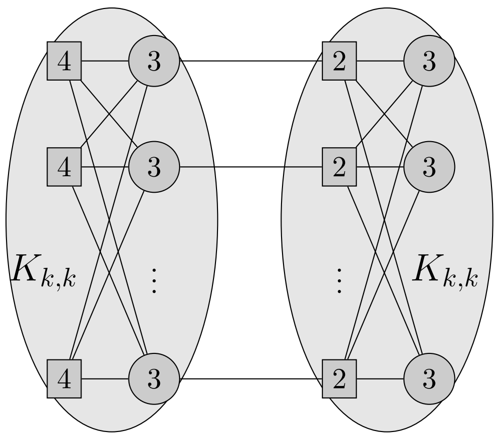
Now for the 2-connected case we can modify the above gadget as follows to retain a new gadget $\mathbb{G}_2$:



As above, we could also build a gadget with a constant degree, but would definitely lose the planarity (which we might have lost in the 2-connected case anyways), because we would have two identical inverted trees T_1, T_2 where we would connect the kth layer of any such tree with the $k + 1$th layer of the other one. I.e. if there is an edge $(a^1_{i,j}, a^1_{i',j+1})$ in T_1, then we would also have to add the edge $(a^1_{i,j}, a^2_{i',j+1})$ from T_1 to T_2.

Removing the vertex v_2 would make the gadget useless as Player 2 could force a cycle between r_1 and r_2. Removing v_1 or v_3 would render the upper respectively the lower half of the gadget useless.

We can generalise the above gadget $\mathbb{G}_2$ as follows. We let $\mathbb{G}_k$ be two copies of the complete bipartite graph $K_{k,k}$ on k 1-vertices and k 2-vertices. The priorities in the first copy are 4 and 3, the ones in the second 2 and 3. Each 1-vertex in the first copy is connected in a 1-1 fashion with a 2-vertex in the second copy. The following picture depicts $\mathbb{G}_k$ (all edges are both ways):

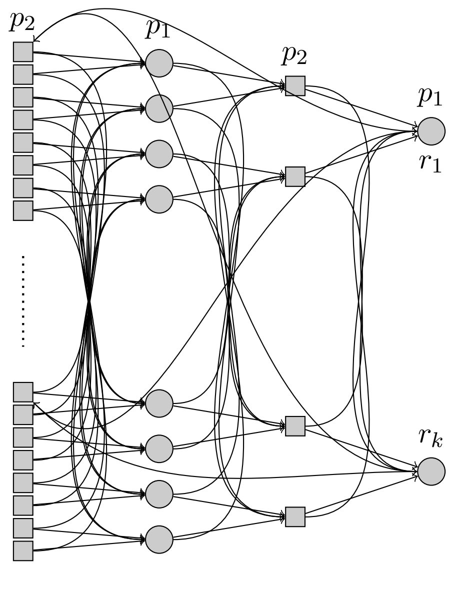


Observation 7 *Let G be any k-connected graph, then there exists a k-connected graph G' such that there is a trivial strategy on G' and removing a single vertex v makes the strategy in G' as hard to compute as in G.*

Proof. Let $G' := G \cup \mathbb{G}_k$, where k distinct 1-vertices $g_1, \ldots, g_k$ (with priority at least 5) from G are connected to k distinct 2-vertices $r_1, \ldots .r_k$ in the first copy of $K_{k,k}$

of G_k. Then removing any 2-vertex v_i in the second copy of $K_{k,k}$ in $\mathbb{G}_k$ means that the associated 1-vertex in the first vertex cannot be used to play into the second copy anymore, so Player 2 can keep the game in the first copy, where he has the highest priority. So, Player 1 will never play into the gadget and his strategy is to be computed from scratch and coincides with the one on G.

As mentioned before we can also construct the gadget using k many identical inverted trees to get a gadget that is k-connected and of constant degree $k+3$. More precisely we take k many identical inverted trees $T_1, \ldots, T_k$ and add to each inner inner vertex $k-1$ outgoing edges in the following fashion. Let $a_{i,h,j}$ denote the j-th vertex in the h-th layer of T_i and let $(a_{i,h,j}, a_{i,h+1,j'}) \in E(T_i)$, then we add the edge $(a_{i,h,j}, a_{i',h+1,j'})$ for any $i' \leq k$. Moreover, in each inverted tree T_i we let an edge go from the root directly to the first leaf $a_{i,1,1}$. We fix an even and an odd priority $p_1 < p_2$ and give alternating priorities to each layer, starting with p_2 for the leafs. We call this gadget with k-many inverted trees, ℓ leafs per tree and alternating priorities $p_1 < p_2$, $\mathbb{T}^k_{\ell,p_1,p_2}$:



Observe that the k-connectivity is established through the adjoined k-connected graph G. Also, we have to require that the lowest priority is not used in the graph G (can always be achieved by renumbering).

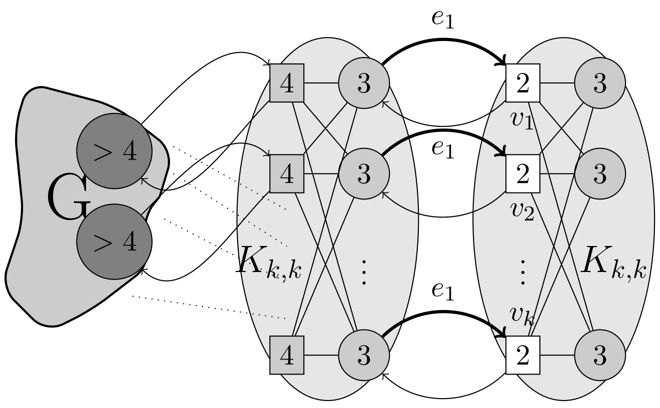
Observation 8 *Let G be any k-connected graph with degree at most κ, then there exists a k-connected graph G' of degree at most* $\max\{k+3, \kappa\}$ *such that there is a trivial strategy on G' and removing at most k vertices $v_1, \ldots, v_k$ the strategy in G' is as hard to compute as in G.*

Proof. As the proof of Observation 7 using the above family of k many inverted trees and then going into $\mathbb{G}_k$ by connecting the roots to the 2-vertices in the left $K_{k,k}$ of $\mathbb{G}_k$.

For the removal of edges or alteration of priorities we can essentially use the same gadget.

Observation 9 *Let G be any k-connected graph, then there exists a k-connected graph G' such that there is a trivial strategy on G' and removing a single edge e_i the strategy in G' is as hard to compute as in G. Moreover altering a single priority p_i the strategy in G' is as hard to compute as in G.*

Proof. We let $G' = G \cup \mathbb{G}_k$ and $p' \restriction G = p + 4$. In addition, we connect each 1-vertex in G with a 2-vertex in the first copy of $K_{k,k}$ in $\mathbb{G}_k$. See the following picture:



To make the gadget a losing move for Player 1, remove any edge e_i or change the priorities of any v_i to 4. The strategy for Player 2 is to play to the 1-vertex adjacent to v_i or that has been incident with e_i. Obviously this is a winning strategy for Player 2, so Player 1 will not play into the gadget and his strategy has to be recomputed.

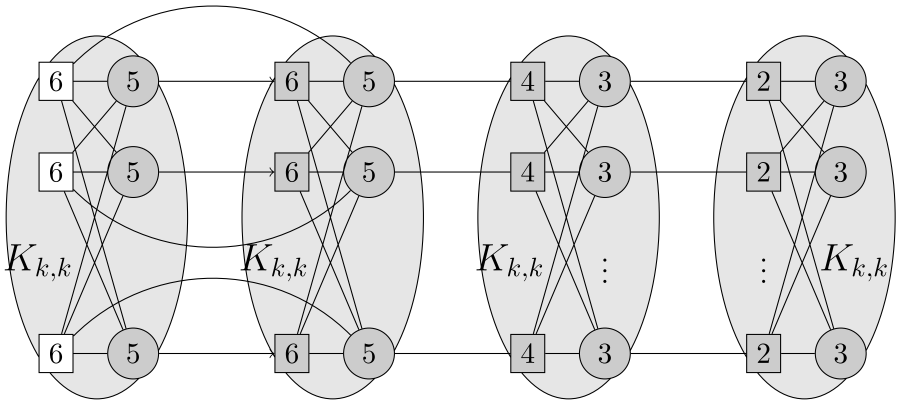
The above gadgets $\mathbb{G}_k$ are still feasible for $f(n)$-connected graphs, for any reasonable choice of f, as they then are of at most quadratic size in the size of the original graph. So we can conclude that removing edges or vertices, or changing priorities is infeasible in the above cases.

We can use the same gadget to show that adding a vertex is infeasible.

Observation 10 *Let G be any k-connected graph, then there exists a k-connected graph G' such that there is a trivial strategy on G' and adding a single vertex v the strategy in G' is as hard to compute as in G.*

Proof. Let $G' = G \cup \mathbb{G}_k$ as above. Then adding a vertex v that establishes a path from a 2-vertex in the second copy of $K_{k,k}$ to one in the first copy clearly makes the gadget a winning region for Player 2.

To show hardness of adding edges, we will basically make the same argument as for adding a vertex. The only problem we run into, is that we have nowhere sensible to go from a 2-vertex, as we would like to end up in the first $K_{k,k}$ of $\mathbb{G}_k$, but the only option would be to go back to a 1-vertex (we restricted edges to connect 1-vertices with 2-vertices and vice versa), which, of course, would be losing. So we have to add another gadget to $\mathbb{G}_k$ (observe that the edges from the first to the second $K_{k,k}$ are directed only from left to right!):

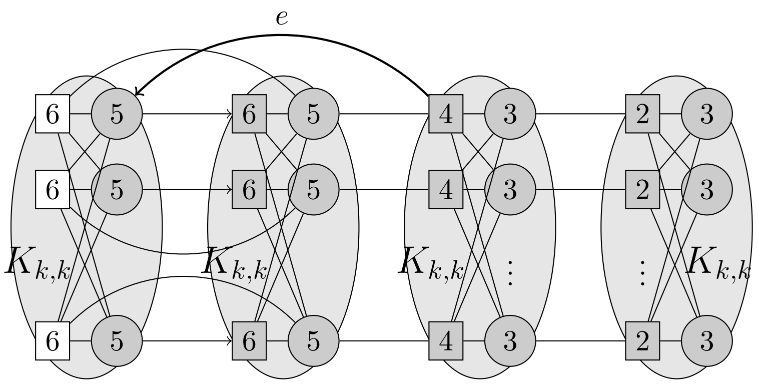


As before we will connect the k-connected graph to the leftmost vertices of this gadget. This gives Player 1 the strategy to play into the gadget and then force the token to the rightmost $K_{k,k}$, where he wins as he has the highest priority.

Observation 11 *Let G be any k-connected graph, then there exists a k-connected graph G' such that there is a trivial strategy on G' and adding a single edge e the strategy in G' is as hard to compute as in G.*

Proof. As mentioned above, G' is the graph G together with the above gadget, granting Player 1 a trivial winning strategy.

However, if we add an edge e to the above gadget as follows, this strategy will be useless.



In the modified gadget, Player 2 can force a cycle on the left half of the gadget and therefore wins for all plays starting there. Thus, Player 1 will not play into the gadget and the complexity of computing the winning regions on $G' \cup \{e\}$ is the same as that of G.

4.2 Planar Graphs

Planar graphs are a very natural class of graphs, but pose one immediate problem when trying to use the argument presented so far. Unless the graph contains no two cycles which intersect in more than one vertex (i.e. unless the graph only has two faces) we cannot connect all vertices belonging to one player with one gadget. The obvious solution is to use several gadgets, at most one for each face, to get similar results as above. This yields the following.

Observation 12 *Given a game graph (G, p), there is a game graph (G', p') extending it, such that there is a trivial strategy on (G', p'), but removing or adding a linear (in the number of faces of G) amount of edges or vertices or altering that many priorities will yield a graph over which winning regions are as hard to compute as on (G, p).*

5 Conclusion

As we have observed above, even for classes of graphs with very strong structural restrictions, it is not possible to do simple alterations to the graph structure while maintaining a reasonable part of the strategy.

It would be very interesting to see how this translates to the random graph. As we need many vertices in our gadgets to have constant degree, these gadgets do not exist in the graph. Therefore, it would be an interesting step to see the influence randomly adding or removing a vertex or an edge to the graph or changing a random priority. Unfortunately, gadgets in the random graph seem to be almost not usable, as all vertices will be connected to various other parts in the graph, which makes the analysis very complicated (and the gadget likely not usable).

6 Acknowledgements

I want to thank Ken-ichi Kawarabayashi for helpful discussions on this topic.

References

1. R. Diestel. Graph Theory. *Graduate Texts in Mathematics*, vol. **173**, Springer, 2000.
2. E. Grädel, W. Thomas and T. Wilke (eds). Automata Logics, and Infinite Games. *Lecture Notes in Computer Science*, vol. **2500**, Springer, 2002.
3. M. Jurdzinski. Deciding the Winner in Parity Games Is in $\mathbf{UP} \cap \mathbf{co} - \mathbf{UP}$. *Information Processing Letters* **68**(3): 119–124, Elsevier 1998.
4. D. Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, vol. **27**: 333–354, 1983.