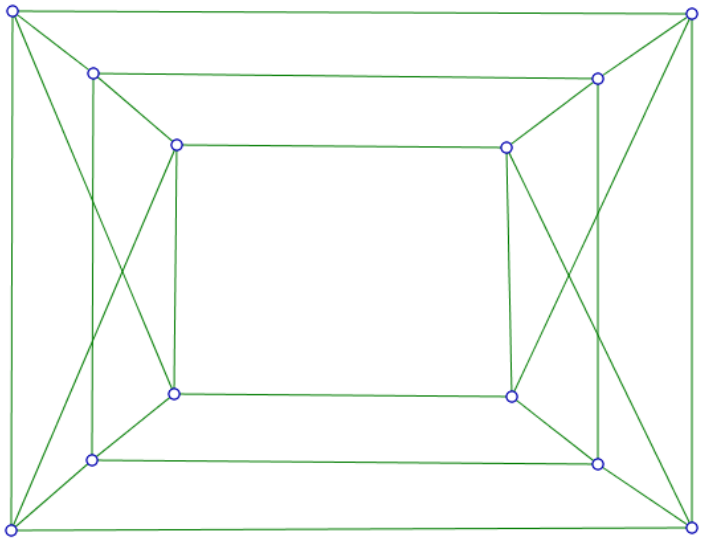
Applications of Convex Optimization on the Graph Isomorphism problem

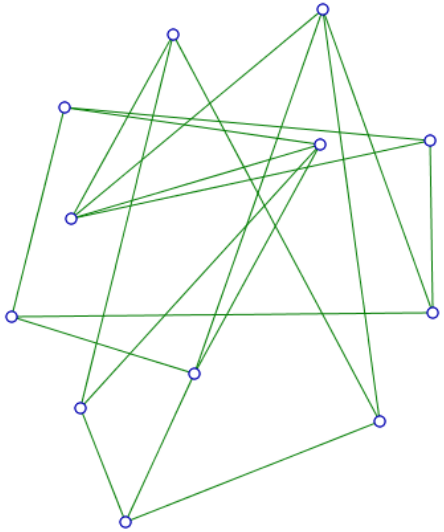
Nicolaos Matsakis[1]

[1]Department of Information and Computer Sciences
UC Irvine

March 8, 2011

Outline





We are asking whether there is a bijection between the sets of vertices of the 2 graphs, such that any two vertices are adjacent in the first graph iff their mapping vertices are also adjacent in the second graph.
Example: Are two chemical substances the same?

Research

- Major open problem in the area of Theory, regarding the specific class that it belongs to.
- Obviously belongs to NP, however it is not, probably, NP-hard. It is also believed that there does not exist any polynomial algorithm for it, making it one of the problems belonging to the class NPI (intermediate problems between NP and P that are not NP-hard). Subraph Isomorphism is a well known NP-hard problem.
- There exist, though, polynomial algorithms for it, regarding the cases of planar graphs, fixed minimum degree graphs, etc

Making the graph a mechanical system

- In nature we can have 2 kind of forces with positive measure: the attractive ones (springs, gravity etc) and the repulsive ones (forces between identical particles etc)
- Spring forces enable the Euclidean distance between the 2 corresponding parts to be in the numerator; the repulsive ones, have that in the denominator.
- Motivation: Why don't we apply forces on a graph and make it a true mechanical system (transforming its edges into springs)? Why don't we regard some sets of its nodes as identical particles (e.g. electrons) and apply repulsive forces between them, making our graph a dynamical system? Or, why don't we do both of them, in order to obtain some information regarding a possible isomorphism between 2 graphs?

Making the graph a mechanical system

- In nature we can have 2 kind of forces with positive measure: the attractive ones (springs, gravity etc) and the repulsive ones (forces between identical particles etc)
- Spring forces enable the Euclidean distance between the 2 corresponding parts to be in the numerator; the repulsive ones, have that in the denominator.
- Motivation: Why don't we apply forces on a graph and make it a true mechanical system (transforming its edges into springs)? Why don't we regard some sets of its nodes as identical particles (e.g. electrons) and apply repulsive forces between them, making our graph a dynamical system? Or, why don't we do both of them, in order to obtain some information regarding a possible isomorphism between 2 graphs?

Making the graph a mechanical system

- In nature we can have 2 kind of forces with positive measure: the attractive ones (springs, gravity etc) and the repulsive ones (forces between identical particles etc)
- Spring forces enable the Euclidean distance between the 2 corresponding parts to be in the numerator; the repulsive ones, have that in the denominator.
- Motivation: Why don't we apply forces on a graph and make it a true mechanical system (transforming its edges into springs)? Why don't we regard some sets of its nodes as identical particles (e.g. electrons) and apply repulsive forces between them, making our graph a dynamical system? Or, why don't we do both of them, in order to obtain some information regarding a possible isomorphism between 2 graphs?

Setting the springs

- The spring force is equal to k*d, where k is the Hooke's constant and d the distance between the 2 sides of the spring minus the rest length of the spring l. The energy associated with a spring force is equal to $\frac{1}{2}k(x-l)^2$.
- It is obvious that 2 graphs that are isomorphic have the same minimum energy associated with the settlement of spring forces. However the total spring energy (assuming k=2) is equal to $\sum_{j=i+1}^{n}\sum_{i=1}^{n}(x_{ij}-l)^2 = x_{ij}^2 - 2x_{ij}l + l^2 = (x_i-x_j)^2+(y_i-y_j)^2 - 2l\sqrt{(x_i-x_j)^2+(y_i-y_j)^2} + l^2$ providing a non-convex (unconstrained) program. So, the computation of the minimum energy cannot be done efficiently and this is due to the square root in the objective function.
- But what about if we could compute the global minimum?

Setting the springs

- The spring force is equal to k*d, where k is the Hooke's constant and d the distance between the 2 sides of the spring minus the rest length of the spring l. The energy associated with a spring force is equal to $\frac{1}{2}k(x-l)^2$.
- It is obvious that 2 graphs that are isomorphic have the same minimum energy associated with the settlement of spring forces. However the total spring energy (assuming k=2) is equal to $\sum_{j=i+1}^{n}\sum_{i=1}^{n}(x_{ij}-l)^2 = x_{ij}^2 - 2x_{ij}l + l^2 = (x_i-x_j)^2+(y_i-y_j)^2 - 2l\sqrt{(x_i-x_j)^2+(y_i-y_j)^2} + l^2$ providing a non-convex (unconstrained) program. So, the computation of the minimum energy cannot be done efficiently and this is due to the square root in the objective function.
- But what about if we could compute the global minimum?

The class of G.I.

- Let's assume that 2 graphs G_1 and G_2 have a different total minimum spring energy. What does this mean about their possible isomorphism?
- Obviously G_1 and G_2 cannot be isomorphic, since they should have equal minimum energies.
- What about if $E(G_1) = E(G_2)$?
- Can we say that the 2 graphs are not isomorphic?
- We can't, as the following example ($k = 2, l = 1$) shows.

The class of G.I.

- Let's assume that 2 graphs G_1 and G_2 have a different total minimum spring energy. What does this mean about their possible isomorphism?
- Obviously G_1 and G_2 cannot be isomorphic, since they should have equal minimum energies.
- What about if $E(G_1) = E(G_2)$?
- Can we say that the 2 graphs are not isomorphic?
- We can't, as the following example ($k = 2, l = 1$) shows.

The class of G.I.

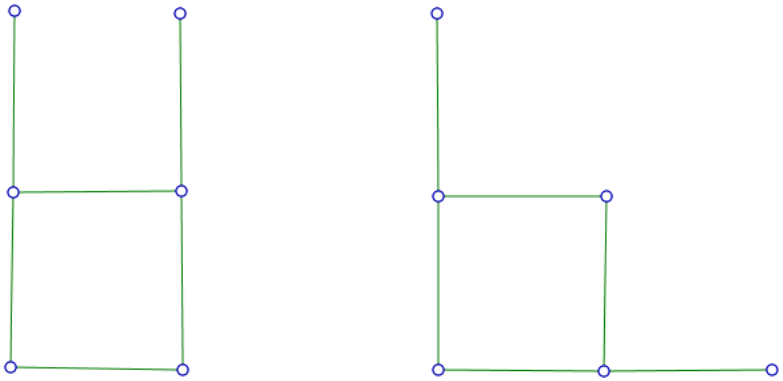
- Let's assume that 2 graphs G_1 and G_2 have a different total minimum spring energy. What does this mean about their possible isomorphism?
- Obviously G_1 and G_2 cannot be isomorphic, since they should have equal minimum energies.
- What about if $E(G_1) = E(G_2)$?
- Can we say that the 2 graphs are not isomorphic?
- We can't, as the following example ($k = 2, l = 1$) shows.

The class of G.I.

- Let's assume that 2 graphs G_1 and G_2 have a different total minimum spring energy. What does this mean about their possible isomorphism?
- Obviously G_1 and G_2 cannot be isomorphic, since they should have equal minimum energies.
- What about if $E(G_1) = E(G_2)$?
- Can we say that the 2 graphs are not isomorphic?
- We can't, as the following example ($k = 2, l = 1$) shows.

A counterexample towards showing $GI \in RP$

- The following 2 graphs have equal energy (each spring is set to unit length) but they are, obviously, not isomorphic.So, -if what we do is correct of course!- the Graph Isomorphism problem should belong to the class co-RP, which is the complement class of problems with polynomial Monte Carlo algorithms. In other words, we may have false positives but never false negatives. co-RP is located between P and co-NP, assuming $P \neq NP$.



Our thoughts on the problem

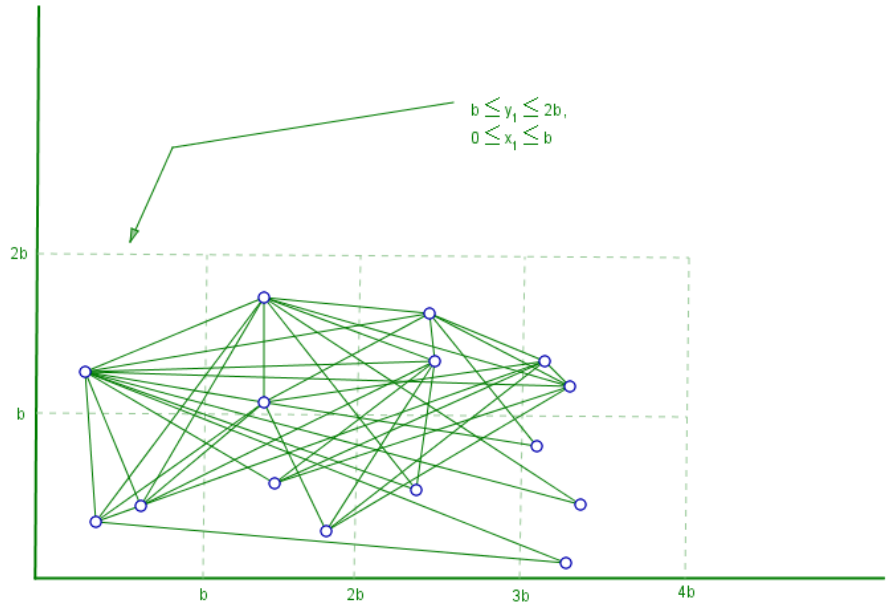
- First of all, we must get rid of the square root of the previously described total energy (objective function). By removing it we obtain a convex quadratic unconstrained program which accounts for the good news. The bad news is that, this specific program has a global minimum at 0. So we do not gain anything by dropping the non-convexity that way.
- The key is to find a way to divide the graph according to some feature of it (e.g using inequalities) that will preserve the fact that we can not get false negatives.
- So, we arrange the graph nodes according to descending degree and we place each node of a specific degree inside a square defined by four inequality constraints.

Our thoughts on the problem

- First of all, we must get rid of the square root of the previously described total energy (objective function). By removing it we obtain a convex quadratic unconstrained program which accounts for the good news. The bad news is that, this specific program has a global minimum at 0. So we do not gain anything by dropping the non-convexity that way.
- The key is to find a way to divide the graph according to some feature of it (e.g using inequalities) that will preserve the fact that we can not get false negatives.
- So, we arrange the graph nodes according to descending degree and we place each node of a specific degree inside a square defined by four inequality constraints.

Example

- In the following picture we can see a graph of 15 nodes (with degrees 11,8,8,7,7,6,6,5,5,4,4,3,2,2,2) set this way by using 32 inequalities setting each kind of node into a specific square and then applying the spring forces.



- By doing that we obtain information on what is going one between nodes of different degrees. We create subgraphs that are almost regular and we may be able to reduce GI to smaller problems regarding isomorphism inside each square.
- However, unfortunately the current best bound for GI for regular graphs is subexponential $2^{O(\sqrt{n}log^2 n)}$, so we must think of something else...
- We thought of applying electrostatic forces (according to the inverse Coulomb square law) between nodes inside each square.
- This solves the problem and gives a total minimum energy which characterizes each graph. However, the presence of the Euclidean distance in the denominator provides highly perplexed non-linear non-convex programs, so we must head our attention to some other way.

- By doing that we obtain information on what is going one between nodes of different degrees. We create subgraphs that are almost regular and we may be able to reduce GI to smaller problems regarding isomorphism inside each square.
- However, unfortunately the current best bound for GI for regular graphs is subexponential $2^{O(\sqrt{n}log^2 n)}$, so we must think of something else...
- We thought of applying electrostatic forces (according to the inverse Coulomb square law) between nodes inside each square.
- This solves the problem and gives a total minimum energy which characterizes each graph. However, the presence of the Euclidean distance in the denominator provides highly perplexed non-linear non-convex programs, so we must head our attention to some other way.

- By doing that we obtain information on what is going one between nodes of different degrees. We create subgraphs that are almost regular and we may be able to reduce GI to smaller problems regarding isomorphism inside each square.
- However, unfortunately the current best bound for GI for regular graphs is subexponential $2^{O(\sqrt{n}log^2 n)}$, so we must think of something else...
- We thought of applying electrostatic forces (according to the inverse Coulomb square law) between nodes inside each square.
- This solves the problem and gives a total minimum energy which characterizes each graph. However, the presence of the Euclidean distance in the denominator provides highly perplexed non-linear non-convex programs, so we must head our attention to some other way.

- By doing that we obtain information on what is going one between nodes of different degrees. We create subgraphs that are almost regular and we may be able to reduce GI to smaller problems regarding isomorphism inside each square.
- However, unfortunately the current best bound for GI for regular graphs is subexponential $2^{O(\sqrt{n}log^2 n)}$, so we must think of something else...
- We thought of applying electrostatic forces (according to the inverse Coulomb square law) between nodes inside each square.
- This solves the problem and gives a total minimum energy which characterizes each graph. However, the presence of the Euclidean distance in the denominator provides highly perplexed non-linear non-convex programs, so we must head our attention to some other way.

Alternative way

- The key problem is that we cannot form convex quadratic inequality constraints and therefore a QCQP, since our inequalities are of the form $(1/2) * x^T P_i x + q_i^T x + r_i \geq 0$, even though the P_i's are positive semidefinite matrices.
- These inequality constraints represent the fact that for a spring with zero rest length, its length cannot become less than a specific (actually any length, but equal for every spring) length, in order to avoid the problematic solution to 0 (collapse of all nodes to a specific point in plane).
- Minimize $(1/2) * x^T P_o x + q_o^T x + r_o$
  subject to $(x_i - x_j)^2 + (y_i - y_j)^2 \geq 1$, for all nodes i,j connected to each other (use adjacency matrix as upper left and lower right block in 2n x 2n matrix, dividing by 2 the $x_i x_j$ coefficients diagonally).
- The number of these constraints is equal to $|E|$.

Alternative way

- The key problem is that we cannot form convex quadratic inequality constraints and therefore a QCQP, since our inequalities are of the form $(1/2) * x^T P_i x + q_i^T x + r_i \geq 0$, even though the P_i's are positive semidefinite matrices.
- These inequality constraints represent the fact that for a spring with zero rest length, its length cannot become less than a specific (actually any length, but equal for every spring) length, in order to avoid the problematic solution to 0 (collapse of all nodes to a specific point in plane).
- Minimize $(1/2) * x^T P_o x + q_o^T x + r_o$
  subject to $(x_i - x_j)^2 + (y_i - y_j)^2 \geq 1$, for all nodes i,j connected to each other (use adjacency matrix as upper left and lower right block in 2n x 2n matrix, dividing by 2 the $x_i x_j$ coefficients diagonally).
- The number of these constraints is equal to $|E|$.

Alternative way

- The key problem is that we cannot form convex quadratic inequality constraints and therefore a QCQP, since our inequalities are of the form $(1/2) * x^T P_i x + q_i^T x + r_i \geq 0$, even though the P_i's are positive semidefinite matrices.
- These inequality constraints represent the fact that for a spring with zero rest length, its length cannot become less than a specific (actually any length, but equal for every spring) length, in order to avoid the problematic solution to 0 (collapse of all nodes to a specific point in plane).
- Minimize $(1/2) * x^T P_o x + q_o^T x + r_o$
  subject to $(x_i - x_j)^2 + (y_i - y_j)^2 \geq 1$, for all nodes i,j connected to each other (use adjacency matrix as upper left and lower right block in 2n x 2n matrix, dividing by 2 the $x_i x_j$ coefficients diagonally).
- The number of these constraints is equal to $|E|$.

Heading towards possible convexity

- The key is to understand that the objective function tries to bring each edge to a position of all-equal edge length. A graph may be able to be drawn on plane using equal edge lengths (it is surely planar in this case), but we may, also, not be able to draw it with all-equal edge lengths (even a great variety of planar graphs cannot be drawn like that). An example is any clique with more than 3 nodes (4-clique is planar).
- But what about non-convexity, even if we fix that?

Heading towards possible convexity

- The key is to understand that the objective function tries to bring each edge to a position of all-equal edge length. A graph may be able to be drawn on plane using equal edge lengths (it is surely planar in this case), but we may, also, not be able to draw it with all-equal edge lengths (even a great variety of planar graphs cannot be drawn like that). An example is any clique with more than 3 nodes (4-clique is planar).
- But what about non-convexity, even if we fix that?

There is some hope...

- There is one specific non-convex quadratic program for which we have strong results :)
- The Single Constraint Quadratic program [Boyd/Vandenberghe appendix B]: Minimize a non-convex quadratic function subject to *one* non-convex quadratic constraint.
- If this program is strictly feasible, then there exists strong duality between that and a Semidefinite Program we fix (solvable efficiently)

There is some hope...

- There is one specific non-convex quadratic program for which we have strong results :)
- The Single Constraint Quadratic program [Boyd/Vandenberghe appendix B]:
  Minimize a non-convex quadratic function
  subject to *one* non-convex quadratic constraint.
- If this program is strictly feasible, then there exists strong duality between that and a Semidefinite Program we fix (solvable efficiently)

There is some hope...

- There is one specific non-convex quadratic program for which we have strong results :)
- The Single Constraint Quadratic program [Boyd/Vandenberghe appendix B]:
  Minimize a non-convex quadratic function
  subject to *one* non-convex quadratic constraint.
- If this program is strictly feasible, then there exists strong duality between that and a Semidefinite Program we fix (solvable efficiently)

Transformation

- But the SCNQP has only one constraint, while we need $|E|$ of them.
- We can add all constraints into one. However we must assure that this move still provides us with a probability of 1 of not obtaining a false negative.
- So we can create the non-convex quadratic program:
  Minimize $x^T A_o' x$,
  subject to $x^T(-A_o)x + |E| \leq 0$
  and then transform it to the equivalent SDP.
- The work is under progress.

Transformation

- But the SCNQP has only one constraint, while we need $|E|$ of them.
- We can add all constraints into one. However we must assure that this move still provides us with a probability of 1 of not obtaining a false negative.
- So we can create the non-convex quadratic program:
  Minimize $x^T A_o' x$,
  subject to $x^T(-A_o)x + |E| \leq 0$
  and then transform it to the equivalent SDP.
- The work is under progress.

Transformation

- But the SCNQP has only one constraint, while we need $|E|$ of them.
- We can add all constraints into one. However we must assure that this move still provides us with a probability of 1 of not obtaining a false negative.
- So we can create the non-convex quadratic program:
  Minimize $x^T A_o' x$,
  subject to $x^T(-A_o)x + |E| \leq 0$
  and then transform it to the equivalent SDP.
- The work is under progress.