A DETERMINISTIC APPROXIMATION ALGORITHM FOR THE DENSEST K-SUBGRAPH PROBLEM

ALAIN BILLIONNET, FRÉDÉRIC ROUPIN

CEDRIC, CNAM-IIE, 18 allée Jean Rostand 91025 Evry cedex, France.

e-mails: {billionnet,roupin}@iie.cnam.fr

Abstract. In the Densest k-Subgraph problem (DSP), we are given an undirected weighted graph $G = (V, E)$ with n vertices $(v_1, ..., v_n)$. We seek to find a subset of k vertices (k belonging to $\{1, ..., n\}$) which maximizes the number of edges which have their two endpoints in the subset. This problem is NP-hard even for bipartite graphs, and no polynomial-time algorithm with a fixed performance guarantee is known for the general case. Several authors have proposed randomized approximation algorithms for particular cases (especially when $k = \frac{n}{c}$, $c > 1$). But derandomization techniques are not easy to apply here because of the cardinality constraint, and can have a high computational cost. In this paper we present a deterministic $\max(d, \frac{8}{9c})$-approximation algorithm for the Densest k-Subgraph Problem (where d is the density of G). The complexity of our algorithm is only the one of linear programming. This result is obtained by using particular optimal solutions of a linear program associated with the classical 0-1 quadratic formulation of DSP.

Keywords: Approximation, Quadratic programming, Linear programming, Complexity.

1. Introduction

Consider an undirected and weighted graph $G = (V, E)$ with n vertices $(v_1, ..., v_n)$, an integer k belonging to $\{1, ..., n\}$, and a non-negative weight w_{ij} on each edge $[v_i, v_j]$. The Heaviest k-Subgraph problem (HSP) consists in determining a subset S of k vertices such that the total edge weight of the subgraph induced by S is maximized. HSP is also known under the name of p-dispersion problem, p-defence-sum problem (Krarup et al. 2002), k-cluster problem, and densest

K-SUBGRAPH PROBLEM (DSP) when all the edge weights are equal to 1. HSP is NP-hard even for unweighted bipartite graphs (using a polynomial reduction from CLIQUE (Corneil and Perl 1984)). For unweighted graphs CLIQUE reduces polynomially to DSP: there is a clique of size k in G if and only if the optimal value of DSP is equal to $\frac{k(k-1)}{2}$. Strong negative results have been obtained about the approximability of the CLIQUE problem. Unfortunately, there is no direct way to use them for the HSP problem.

Recall that a r-approximation algorithm for a maximization problem (Π) returns for any instance I of (Π) a feasible solution of value v_a such that $\frac{v_a}{v^*} \geq r$, where v^* is the optimal value of I. The parameter $0 \leq r \leq 1$ is known as the *approximation ratio* of the algorithm.

Many approximation results are known for HSP (Kortsarz and Peleg 1993, Asahiro et al. 2000, Feige et al. 2001b, Jäger and Srivastav 2005) but no approximation algorithm with fixed ratio-bound have been found to date and the question of knowing if such an algorithm exists is open. However, HSP admits an approximation ratio of at least $\max\left(k/2n,\, n^{\varepsilon-1/3}\right)$ for all k and some $\varepsilon > 0$ (Kortsarz and Peleg 1993, Feige et al. 2001a). A well-known greedy heuristic (Asahiro et al. 2000) returns a feasible solution of value v^a, and provides upper and lower bounds on the worst case approximation ratio $R = \frac{1}{r}$ of v^* upon v^a:

$$\left(\frac{1}{2} + \frac{n}{2k}\right)^2 - O\left(n^{-\frac{1}{3}}\right) \leq R \leq \left(\frac{1}{2} + \frac{n}{2k}\right)^2 + O\left(\frac{1}{n}\right) \text{ for } \frac{n}{3} < k \leq n$$

and

$$2\left(\frac{n}{k} - 1\right) - O\left(\frac{1}{k}\right) \leq R \leq 2\left(\frac{n}{k} - 1\right) + O\left(\frac{n}{k^2}\right) \text{ for } k \leq \frac{n}{3}$$

Let $c = \frac{n}{k}$. For $c = 2$, R is less than or equal to $\frac{9}{4} + O\left(\frac{1}{n}\right)$, for $c = 3$, we obtain $4 + O\left(\frac{n}{k^2}\right)$, and for $c = 4$, we get $6 + O\left(\frac{n}{k^2}\right)$. For some particular cases, there are better approximation results. For HSP, when the weights satisfy the triangle inequality, there exists a $\frac{1}{2}$-approximation algorithm (Hassin et al. 1997). All the previous results for HSP obviously apply to DSP (the DENSEST k-SUBGRAPH PROBLEM).

On the other hand, many randomized algorithms have been proposed for HSP and DSP. Some of them use the optimal solution of the natural linear relaxation of a quadratic formulation of DSP (Han et al. 2000). Others are based on semidefinite relaxations (Feige et al. 2001b, Han et al. 2002, Jäger and Srivastav 2005). Unfortunately, to the best of our knowledge, no derandomization has been explicitly presented for these algorithms. Note that the task is complicated here because of the cardinality constraint. For such problems, as noted in (Feige et al. 2001c) for the max-bisection problem, the impact on the worst-case ratio by "repairing" the solution is very difficult to analyse. In (Arora et al. 1999), starting from a 0-1 quadratic formulation, such a successful analysis leads to a polynomial time approximation scheme (PTAS) for DSP in everywhere-dense graphs (the minimum degree in G is $\Omega(n)$). This last result holds when G is a dense-graph (the number of edges is $\Omega\left(n^2\right)$) and $k = \Omega\left(n\right)$. But these hypothesis on the density are crucial.

Our contribution is to propose a simple deterministic combinatorial algorithm which uses as input an optimal solution of a linear relaxation of a quadratic formulation of DSP. The total complexity of our algorithm is only the one of linear programming, and its worst case approximation ratio is $\max\left(d, \frac{8}{9c}\right)$ where $d = \frac{2|E|}{n(n-1)}$ is the density of G. Indeed, we did not use derandomization techniques which are not easy to apply here, and can increase dramatically the overall complexity. In Section 2, we recall the randomized approaches based on linear and semidefinite programming for HSP and DSP. In Section 3, we present our deterministic algorithm. First, we characterize some optimal solutions of a classical linear relaxation of the 0-1 quadratic formulation of DSP. Second, we compare the value of these solutions for the linear objective function and for the quadratic objective function. Moreover, we study the integrality gap in the quadratic formulation. Using these results, we design a new deterministic approximation algorithm for DSP. In Section 4, we apply our algorithm to a small instance of DSP, then we compare our worst-case ratio with previous ones. Section 5 is a conclusion.

2. Randomized approaches

First, note that in DSP the probability to have an edge between two vertices is $d = \frac{2|E|}{n(n-1)}$, thus the expectation of the number of edges of a subgraph with k vertices in G equals $d\frac{k(k-1)}{2}$. For HSP, there is a simple deterministic $O\left(n^2\right)$ algorithm which achieves the expected value of a random solution. This greedy heuristic is used for instance in (Asahiro et al. 2000). Just remove one by one $n-k$ vertices from G by choosing at each step i the vertex v_i such that $\sum_{[v_i,v_i]\in E} w_{ij}$ is minimal (i.e. the minimum degree for DSP) in the remaining graph. It can be seen as a deterministic version of the simplest randomized algorithm for HSP. Using this greedy procedure, one can prove that (Srivastav and Wolf 1998):

Theorem 2.1. *The HSP problem in a weighted graph of n vertices has a $\frac{k(k-1)}{n(n-1)}$-approximation polynomial-time algorithm.*

Corollary 2.2. *The DSP problem in a graph of n vertices has a d-approximation polynomial time algorithm, where $d = \frac{2|E|}{n(n-1)}$ is the density of G.*

Now, we recall more sophisticated randomized approaches and discuss derandomization issues. For HSP, in (Feige et al. 2001b) the authors prove that the ratio associated to the randomized algorithm proposed in (Srivastav and Wolf 1998) is incorrect for the particular semidefinite program proposed in this last paper. But by following similar ideas, they present a randomized approximation algorithm using a tighter semidefinite relaxation. The exact expression of their ratio uses constants derived in the approximation algorithms in (Goemans and Williamson 1995) for MAX-k-CUT and MAX-DICUT. This ratio was the first to beat the target $\frac{1}{c}$ when c is close to 2. Semidefinite relaxations for HSP have been also proposed in (Han et al. 2002) by adding a new constraint to the semidefinite program given in (Srivastav and Wolf 1998), and more recently in (Jäger and Srivastav 2005), where worst-case ratios have been improved for $0.04 < c < 0.68$. Unfortunately, all these approaches imply to solve a semidefinite program and to be able to derandomize the algorithm by using conditional probabilities (Srivastav 2001) which is a polynomial process but has a very high computational cost here because of the random hyperplane

approach. For instance, in (Hariharan and Mahajan 1999), authors report a complexity of $O\left(n^{30}\right)$ for the 3-Vertex Coloring problem. Moreover, for our problem one has still to repair the final solution to satisfy the cardinality constraint. In (Feige et al. 2001b), this is done when c is close to 2 by using the greedy heuristic of Theorem 2.1. The same difficulty arises in the linear programming approach when one uses a rounding randomized algorithm. Indeed, consider the following formulation of DSP:

$$(P)\left\{\max\ f(x)=\sum_{i<j,\,[v_i,v_j]\in E} x_i x_j\ :\ \sum_{i=1}^{n} x_i=k,\ x\in\{0,1\}^n\right\}$$

where $x_i = 1$ if and only if vertex v_i is put into the subset. Let $(\underline{P})$ be the continuous relaxation of (P), obtained by replacing the constraint $x \in \{0,1\}^n$ by $x \in [0,1]^n$. Note that (P) can be formulated as:

$$(PL)\left\{\max\ f_L(x)=\sum_{i<j,\,[v_i,v_j]\in E} \min\left(x_i,x_j\right)\ :\ \sum_{i=1}^{n} x_i=k,\ x\in\{0,1\}^n\right\}$$

It is easy to formulate $(\underline{PL})$, the continuous relaxation of (PL) ($x \in [0,1]^n$ in place of $x \in \{0,1\}^n$), as a linear program by adding $|E|$ new variables z_{ij} and $2\,|E|$ constraints $z_{ij} \leq x_i$ and $z_{ij} \leq x_j$.

A randomized algorithm based on $(\underline{PL})$ is cited in (Feige et al. 2001a) and attributed to Goemans. Its expected worst-case ratio equals $\frac{1}{c}$, and it can be roughly stated as follows. First, consider (x', z'), an optimal solution of $(\underline{PL})$. Second, randomly round the n fractional components of x' to 1 with probability $\sqrt{x'_i}$. The probability that an edge $[v_i, v_j]$ is selected is at least z'_{ij}. Thus the expected number of edges is at least the optimal value of $(\underline{PL})$. One can verify that the expected number of vertices is less that $\sqrt{nk}$. Getting from $\sqrt{nk}$ vertices to k (by randomly choosing k vertices) looses $\frac{k^2}{(\sqrt{nk})^2}$ in the ratio, namely, gives $\frac{1}{c}$ ratio.

Note that there are two randomized steps in this algorithm. A derandomization for the first one (randomized rounding) can be achieved by using the results presented in the seminal work of (Raghavan and Thompson 1987, Raghavan 1988). The 0-1 solution obtained here will be such that $\sum_i x_i \leq \sqrt{nk} + O(\sqrt{nlog(n)})$ with

high probability. In (Han et al. 2000) the authors do not derandomize the initial step, but they propose to choose the final k vertices greedily by using Theorem 2.1 (instead of taking them randomly). This leads to an approximation ratio of $\left(\frac{1}{1+k^{-\frac{1}{3}}}\right)^2\left(\frac{1}{c}-\frac{1}{2}n^2e^{-\frac{n^{\frac{1}{9}}}{3}}\right)$. Thus even for n sufficiently large, it looses a factor $\left(\frac{1}{1+k^{-\frac{1}{3}}}\right)^2$ in the ratio $\frac{1}{c}$. For instance, to get a approximation ratio equal to $\frac{0.95}{c}$ then one must have $\left(\frac{1}{1+k^{-\frac{1}{3}}}\right)^2 \geq 0.95$ and thus k must be larger than 57038. Moreover, the additional error $O(\sqrt{nlog(n)})$ for the cardinality constraint is not taken into account since the initial randomized rounding step is not derandomized. All these results illustrate the difficulty to obtain efficient deterministic versions with the same worst case ratio of randomized algorithms for HSP and DSP.

3. A simple deterministic algorithm to approximate DSP

In this section, we assume that the graph $G = (V, E)$ is unweighted and that $k \geq 3$ since otherwise the optimal solution of DSP is trivial.

The main result (Theorem 3.5) is obtained by starting from an optimal solution x^L of ($\underline{PL}$). First, we build $x^{L\beta}$, a particular optimal solution of ($\underline{PL}$) (Lemma 3.1). Second, starting from $x^{L\beta}$ we obtain x^Q, a particular feasible solution of ($\underline{P}$), and finally, x^P, a feasible solution of (P) (formulation of DSP). At each step, we are able to bound the gap between the values of these different solutions. The complexity of our approximation algorithm is only the one of linear programming, because the others steps can be implemented in $O\left(n^2\right)$ and no derandomization is required. Moreover, our worst-case ratio ($\max\left(d, \frac{8}{9c}\right)$) is valid for any k.

To start, we are going to characterize some optimal solutions of a special mathematical program in order to obtain particular optimal solutions of ($\underline{PL}$). Consider $G = (V, E)$, a graph with n vertices $\{v_1, ..., v_n\}$, and (Π), the following continuous linear program:

$$(\Pi)\ \left\{ \text{Max } h(x) = \sum_{i=1}^{n} C_i x_i + \sum_{i<j,\, [v_i,v_j]\in E} \min\left(x_i, x_j\right) \ :\ \sum_{i=1}^{n} x_i = k,\ x \in [0,1]^n \right\}$$

where C_i for all i in $\{1, ..., n\}$ is a non-negative coefficient.

Lemma 3.1. *Let x^* be an optimal solution of* (Π). *If $0 < x_i^* < 1$ for all $i \in \{1, ..., n\}$ then $x'_i = \frac{k}{n}$ $(i = 1, ..., n)$ is also an optimal solution of* (Π).

Proof. Consider

$$\begin{aligned}\Delta &= h(x') - h(x^*) \\ &= \sum_{i=1}^{n} C_i \left(\frac{k}{n} - x_i^*\right) + \sum_{i<j,\,[v_i,v_j]\in E} \left(\frac{k}{n} - \min\left(x_i^*, x_j^*\right)\right)\end{aligned}$$

Since x^* is an optimal solution of (Π) we have $\Delta \leq 0$. Now consider x^o, a feasible solution of (Π), defined by $x_i^o = x_i^* + \rho\left(x_i^* - \frac{k}{n}\right)$ $(i = 1, ..., n)$, where ρ is a positive real such that $0 < x_i^* + \rho\left(x_i^* - \frac{k}{n}\right) < 1$ (ρ exists since $0 < x_i^* < 1$). We get

$$\begin{aligned}h(x^o) - h(x^*) &= \sum_{i<j,\,[v_i,v_j]\in E} \min\left(x_i^* + \rho\left(x_i^* - \frac{k}{n}\right),\, x_j^* + \rho\left(x_j^* - \frac{k}{n}\right)\right) \\ &\quad - \sum_{i<j,\,[v_i,v_j]\in E} \min\left(x_i^*,\, x_j^*\right) \\ &\quad + \sum_{i=1}^{n} C_i \rho\left(x_i^* - \frac{k}{n}\right)\end{aligned}$$

By examining the two cases $x_i^* \leq x_j^*$ and $x_i^* > x_j^*$, one can see that $h(x^o) - h(x^*) = -\rho\Delta$. Since (x^*) is optimal , $-\Delta \leq 0$ and finally $\Delta = 0$; this implies the optimality of (x'). □

A consequence of Lemma 3.1 is that from any optimal solution of $(\underline{PL})$, one can obtain another optimal solution whose fractional components are all equal.

Lemma 3.2. *For DSP, if $x^{L\beta}$ is a feasible solution of $(\underline{PL})$ whose fractional components are all equal to the same value β then $f\left(x^{L\beta}\right) \geq \frac{k}{n} f_L\left(x^{L\beta}\right)$.*

Proof. We define $X_0 = \left\{i \mid x_i^{L\beta} = 0\right\}$, $X_1 = \left\{i \mid x_i^{L\beta} = 1\right\}$, $X_\beta = \left\{i \mid x_i^{L\beta} = \beta\right\}$,
$E_{1,\beta} = \left\{i, j \mid [v_i, v_j] \in E\,;\, (x_i^{L\beta} = 1,\, x_j^{L\beta} = \beta) \text{ or } (x_i^{L\beta} = \beta,\, x_j^{L\beta} = 1)\right\}$,
$E_{\beta,\beta} = \{i, j \mid [v_i, v_j] \in E\,;\, x_i^{L\beta} = \beta,\, x_j^{L\beta} = \beta\}$,
and $E_{1,1} = \{i, j \mid [v_i, v_j] \in E\,;\, x_i^{L\beta} = 1,\, x_j^{L\beta} = 1\}$. Let n_0, n_1, n_β, $e_{1,\beta}$, $e_{\beta,\beta}$, $e_{1,1}$ be respectively the cardinality of X_0, X_1, X_β, $E_{1,\beta}$, $E_{\beta,\beta}$ and $E_{1,1}$. Here we get $\beta = \frac{k-n_1}{n-n_1-n_0}$ (since $\sum_{i=1}^{n} x_i^{L\beta} = k$ and all fractional components are equal). First, if $n_1 = 0$, one can see that $f\left(x^{L\beta}\right) = \frac{k}{n-n_0} f_L\left(x^{L\beta}\right) \geq \frac{k}{n} f_L\left(x^{L\beta}\right)$ (for all (i, j), $x_i^{L\beta} x_j^{L\beta} = \beta \min(x_i^{L\beta}, x_j^{L\beta})$ since $x_i^{L\beta} \in \{0, \beta\}$). Otherwise, consider the

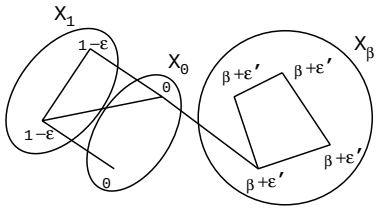

FIGURE 1. $\left(x^0\right)$ is built from $\left(x^{L\beta}\right)$

following feasible solution (x^o) of $(\underline{PL})$ (see Figure 1) : $x_i^o = x_i^{L\beta} - \varepsilon$ $(i \in X_1)$, $x_i^o = x_i^{L\beta}$ $(i \in X_0)$, $x_i^o = x_i^{L\beta} + \varepsilon'$ $(i \in X_\beta)$, where ε and ε' are two positive real numbers satisfying $\varepsilon + \varepsilon' \leq 1 - \beta$ (to have $1 - \varepsilon > \beta + \varepsilon'$, this implies $\varepsilon \leq 1$ and $\varepsilon' \leq 1 - \beta$), and $\varepsilon n_1 = \varepsilon' n_\beta$ (to satisfy constraint $\sum_{i=1}^n x_i^o = k$). Note it is always possible to find such numbers by taking them small enough. We have $h\left(x^o\right) - h\left(x^{L\beta}\right) = e_{\beta,\beta}\varepsilon' - e_{1,1}\varepsilon + e_{1,\beta}\varepsilon' \leq 0$ which implies $e_{\beta,\beta}\frac{n_1}{n_\beta}\varepsilon - e_{1,1}\varepsilon \leq 0$, thus $e_{1,1} \geq e_{\beta,\beta}\frac{n_1}{n_\beta}$ (recall that h is the objective function in (Π)). We know that $f\left(x^{L\beta}\right) = e_{\beta,\beta}\beta^2 + e_{1,1} + e_{1,\beta}\beta$, and $f_L\left(x^{L\beta}\right) = e_{\beta,\beta}\beta + e_{1,1} + e_{1,\beta}\beta$. Thus we get

$$\begin{aligned} \frac{f\left(x^{L\beta}\right)}{f_L\left(x^{L\beta}\right)} &\geq \frac{e_{1,1} + e_{\beta,\beta}\beta^2}{e_{1,1} + e_{\beta,\beta}\beta} \\ &\geq \frac{\frac{n_1}{n_\beta} + \beta^2}{\frac{n_1}{n_\beta} + \beta} = \frac{n_1 + \frac{(k-n_1)^2}{n_\beta}}{k} \end{aligned}$$

(recall that $\beta = \frac{k-n_1}{n_\beta}$). Note that

$$\frac{n_1 + \frac{(k-n_1)^2}{n_\beta}}{k} - \frac{k}{n_1 + n_\beta} = \frac{\left(k - (n_1 + n_\beta)\right)^2}{\frac{n_\beta}{n_1}k(n_1 + n_\beta)} \geq 0$$

yielding $\frac{f\left(x^{L\beta}\right)}{f_L\left(x^{L\beta}\right)} \geq \frac{k}{n_1+n_\beta} \geq \frac{k}{n}$. □

In Algorithm 1, we build from any feasible solution x of $(\underline{P})$, a particular feasible solution x^Q of $(\underline{P})$ such that $f\left(x^Q\right) \geq f(x)$. Then we obtain in Algorithm 2 a feasible solution x^P of (P) from x^Q. Finally we prove in Lemma 3.3 that the gap between $f\left(x^Q\right)$ and $f\left(x^P\right)$ can be bounded.

Algorithm 1 Let x be any feasible solution of $(\underline{P})$. Consider two vertices v_i and v_j of G such that there is no edge between them, and $x_i = \alpha$, $x_j = \beta$ where $0 < \alpha < 1$ and $0 < \beta < 1$ (if such a pair of vertices does not exist then stop).

Consider the following program obtained from $(\underline{P})$ by fixing all the variables x_r $(r = 1, ...n\,;\, r \neq i, j)$ to their values in the feasible solution x.

$$\begin{cases} \max & L(x_i, x_j) = Ax_i + Bx_j + C \\ \text{s.t.} & x_i + x_j = \alpha + \beta \\ & x_i \in [0,1]\,,\; x_j \in [0,1] \end{cases}$$

where A, B and C are positive constants depending on the x_r's values $(r = 1, ..., n\,;\, r \neq i, j)$. This linear program has an obvious optimal solution. Indeed, if $A \geq B$ then $(\min(1,\, \alpha+\beta)\,,\, \beta - \min(1,\, \alpha+\beta) + \alpha)$ is optimal, and if $A < B$ then $(\alpha - \min(1,\, \alpha+\beta) + \beta,\, \min(1,\, \alpha+\beta))$ is optimal. In both cases, at least one variable becomes an integer. By iterating this procedure at most n times, we obtain V_1, V_2, and a feasible solution x^Q of $(\underline{P})$ such that (see Figure 2) :

- $x_i^Q = 0$ or 1 for all i in V_1.
- $0 < x_i^Q < 1$ for all i in V_2.
- The edge $[v_i, v_j]$ belongs to E for all (i, j) in V_2^2.
- $V_1 \subseteq V$, $V_2 \subseteq V$, $V_1 \cup V_2 = V$, and $V_1 \cap V_2 = \emptyset$.

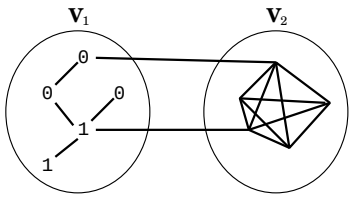

FIGURE 2. *Particular feasible solution* x^Q *of* $(\underline{P})$ *such that* $f\left(x^Q\right) \geq f(x)$

Notation 1 p and q are the respective values of $\mid V_1 \mid$ and $\mid V_2 \mid$, and s is the number of variables set to one in V_1. Moreover, we define

$T_{1\times1} = \left\{(i,j) \in V_1^2 : i < j,\, [v_i, v_j] \in E\right\}$,

$T_{2\times2} = \left\{(i,j) \in V_2^2 : i < j,\, [v_i, v_j] \in E\right\}$,

and $T_{1\times2} = \{(i,j) \in V_1 \times V_2 : i < j,\, [v_i, v_j] \in E\}$.

Algorithm 2 Let x be any feasible solution of $(\underline{P})$. Consider the particular feasible solution x^Q of $(\underline{P})$ obtained from x by Algorithm 1, and build x^P, a feasible solution of (P), by the following way:

We can write that

$$f\left(x^Q\right) = \sum_{(i,j)\in T_{1\times 1}} x_i^Q x_j^Q + \sum_{(i,j)\in T_{1\times 2}} x_i^Q x_j^Q + \sum_{(i,j)\in T_{2\times 2}} x_i^Q x_j^Q$$

Now, fix to their current values all the variables of V_1, and consider the following restricted program

$$\begin{cases} \max & \sum_{i\in V_2} C_i x_i + \sum_{(i,j)\in T_{2\times 2}} x_i x_j + \sum_{(i,j)\in T_{1\times 1}} x_i^Q x_j^Q \\ \text{s.t.} & \sum_{i\in V_2} x_i = k - s \\ & x_i \in \{0,1\}, \forall i \in V_2 \end{cases}$$

where we have $C_i = \sum_{j\in\Gamma(i)\cap V_1} x_i^Q$. The optimal value is clearly equal to

$$\sum_{i\in V_{2[k-s]}} C_i + \frac{(k-s)(k-s-1)}{2} + \sum_{(i,j)\in T_{1\times 1}} x_i^Q x_j^Q$$

where $V_2[k-s]$ is the set of the indices of the $k-s$ variables x_i ($i \in V_2$) having the $k-s$ greatest C_i. Thus an optimal solution is $x_i = 1$ for i in $V_2[k-s]$ and $x_i = 0$ otherwise. Finally, the feasible solution x^P of (P) is given by $x_i^P = x_i^Q$ for i in V_1, $x_i^P = 1$ for i in $V_2[k-s]$, and $x_i^P = 0$ otherwise.

Lemma 3.3. *Let x be any feasible solution of $(\underline{P})$. One can obtain in polynomial-time a feasible solution x^P of (P) (DSP: formulation of HSP in unweighted graphs) such that $f(x) - f\left(x^P\right) \leq \frac{q}{8}$, where q is obtained by applying Algorithm 1 to x.*

Proof. First, we use Algorithms 1 and 2 to obtain x^Q and x^P from x. Recall that $\sum_{i\in V_2} x_i^Q = k - s$. We have:

$$f\left(x^Q\right) - f\left(x^P\right) = \sum_{(i,j)\in T_{1\times 2}} x_i^Q x_j^Q + \sum_{(i,j)\in T_{2\times 2}} x_i^Q x_j^Q - \sum_{i\in V_2[k-s]} C_i - \frac{(k-s)(k-s-1)}{2}$$

Note that $\sum_{(i,j)\in T_{2\times 2}} x_i^Q x_j^Q \leq \left(\frac{k-s}{q}\right)^2 \frac{q(q-1)}{2}$ **[A]**
and $\sum_{i\in V_2[k-s]} C_i \geq \sum_{(i,j)\in T_{1\times 2}} x_i^Q x_j^Q$ **[B]**.
Thus we can write:

$$f\left(x^Q\right) - f\left(x^P\right) \leq \left(\frac{k-s}{q}\right)^2 \frac{q(q-1)}{2} - \frac{(k-s)(k-s-1)}{2}$$

Let $x = k - s$. The maximum value of $\frac{(q-1)}{2q}x^2 - \frac{x(x-1)}{2}$ as a function of x is $\frac{q}{8}$ (obtained in $x = \frac{q}{2}$), thus $f\left(x^Q\right) - f\left(x^P\right) \leq \frac{q}{8}$. □

Corollary 3.4. *Let x be any feasible solution of $(\underline{P})$ of value greater than 1. One can obtain in polynomial-time a feasible solution x^P of (P) (DSP: formulation of HSP in unweighted graphs) such that $\frac{f(x^P)}{f(x)} \geq \frac{1}{2}$.*

Proof. First, we use Algorithms 1 and 2 to obtain x^Q and x^P from x. Since we have $f\left(x^Q\right) \geq f\left(x\right)$, to obtain the claimed result we can prove that $\rho = \frac{f(x^P)}{f(x^Q)} \geq \frac{1}{2}$. We have:

$$\rho = \frac{a + b + \frac{(k-s)(k-s-1)}{2}}{a + \sum_{(i,j)\in T_{1\times 2}} x_i^Q x_j^Q + \sum_{(i,j)\in T_{2\times 2}} x_i^Q x_j^Q}$$

where $a = \sum_{(i,j)\in T_{1\times 1}} x_i^Q x_j^Q$ and $b = \sum_{i\in V_2[k-s]} C_i$. By using **[A]** and **[B]** we obtain:

$$\rho \geq \frac{a + b + \frac{(k-s)(k-s-1)}{2}}{a + b + \left(\frac{k-s}{q}\right)^2 \frac{q(q-1)}{2}} = \frac{a + b + \frac{(k-s)(k-s-1)}{2}}{a + b + \frac{(k-s)^2}{q}\frac{(q-1)}{2}}$$

and thus $\rho \geq \frac{\frac{(k-s)(k-s-1)}{2}}{\frac{(k-s)^2}{q}\frac{(q-1)}{2}} \geq \frac{(k-s-1)q}{(k-s)(q-1)}$. If $k-s \geq 2$ then $\rho \geq \frac{1}{2}$, and if $k-s = 1$ then $f\left(x^P\right) \geq f\left(x^Q\right) - \left(\frac{1}{q}\right)^2 \frac{q(q-1)}{2}$ thus $f\left(x^P\right) \geq f\left(x^Q\right) - \frac{q-1}{2q}$, and finally $f\left(x^P\right) \geq f\left(x^Q\right) - \frac{1}{2}$. Since we have $f\left(x^Q\right) \geq 1$, we get $\rho \geq \frac{1}{2}$. □

One direct consequence of Corollary 3.4 is that the continuous relaxation $(\underline{P})$ is as hard to approximate as (P): if one admits an ε-approximation algorithm so does the other.

Now, we are going to apply Lemma 3.1, 3.2, and 3.3 to obtain a deterministic $\frac{8}{9c}$-approximation algorithm for DSP, the unweighted case of HSP, where $c = \frac{n}{k}$.

Algorithm 3 Let $G = (V, E)$ be an unweighted graph of n vertices, $k \geq 3$, and $n = ck$. Consider the following algorithm which builds a feasible solution x^P of (P):

(1) Solve $(\underline{PL})$ to obtain x^L.

(2) Build $x^{L\beta}$ from x^L by using Lemma 3.1.

(3) Obtain x^Q from $x^{L\beta}$ by using Algorithm 1.

(4) Obtain x^P from x^Q by using Algorithm 2.

Recall that q is equal to the number of fractional components of x^Q (see Notation 1, Algorithm 1).

Theorem 3.5. *If $n = ck$ then DSP admits a deterministic polynomial approximation algorithm that gives a feasible solution x^P of (P) such that $f(x^*) \leq c\left(f\left(x^P\right) + \frac{q}{8}\right)$, where x^* is an optimal solution of (P). Moreover, the complexity of this algorithm is the one of linear programming.*

Proof. We apply Algorithm 3 to (P). We have $f_L\left(x^L\right) \geq f(x^*)$. $x^{L\beta}$ is an optimal solution of $(\underline{PL})$ whose all fractional components are equal (Lemma 3.1). Considering that $x^{L\beta}$ is a feasible solution of $(\underline{P})$, we get $f\left(x^{L\beta}\right) \geq \frac{1}{c} f_L\left(x^{L\beta}\right)$ (Lemma 3.2). Finally, we have $f\left(x^Q\right) \geq f\left(x^{L\beta}\right)$ (Algorithm 1) and $f\left(x^Q\right) - f\left(x^P\right) \leq \frac{q}{8}$ (Lemma 3.3). Thus x^P is a feasible solution for (P) (Algorithm 2) such that $f(x^*) \leq c\left(f\left(x^P\right) + \frac{q}{8}\right)$. Algorithms 1 and 2 can be implemented in $O\left(n^2\right)$, and thus the bottleneck complexity of Algorithm 3 is solving $(\underline{PL})$. □

Corollary 3.6. *If $n = ck$ then DSP admits a polynomial approximation algorithm that gives a feasible solution x^a of (P) such that $\frac{f(x^*)}{f(x_a)} \leq \frac{9}{8}c$, where x^* is an optimal solution of (P).*

Proof. We can assume that $2 \leq q \leq k-1$. Indeed $q \neq 1$ because $\sum_{i=1}^{n} x_i = k$ (there are at least two fractional components or none), if $q \geq k$ then we have already an optimal solution (a clique of size k), and obviously if $q = 0$ then $f(x^*) \leq cf\left(x^P\right)$. Recall that Theorem 3.5 implies $f(x^*) \leq c\left(f\left(x^P\right) + \frac{q}{8}\right)$ **[C]** , where x^P is the feasible solution obtained by Algorithm 3. Now, we consider two cases:

First, if $f\left(x^P\right) \geq q$ then **[C]** implies $\frac{f(x^*)}{f(x^P)} \leq \frac{9}{8}c$, and we set $x^a = x^P$.

Second, if $f\left(x^P\right) \leq q - 1$ **[D]** then **[C]** implies $f(x^*) \leq c\left(q - 1 + \frac{q}{8}\right)$. Now consider two sub-cases:

* If $q \geq 3$ then we consider x^a the feasible solution consisting of the q vertices of the clique completed by $k - q$ arbitrary vertices. We obtain:

$$\frac{f(x^*)}{f(x_a)} \leq \frac{2c\left(q - 1 + \frac{q}{8}\right)}{q(q-1)} \leq \frac{2c\left(\frac{9}{8}q - 1\right)}{q(q-1)} \leq \frac{2c\frac{9}{8}q}{q(q-1)} \leq \frac{9}{8}c$$

* If $q = 2$ then **[D]** and **[C]** imply $f(x^*) \leq c\left(q - 1 + \frac{2}{8}\right) = \frac{5}{4}c$ **[E]**. If there exist two adjacent edges in G then $(k \geq 3)$ we consider as x^a any feasible solution of

value 2, and thus **[E]** implies $\frac{f(x^*)}{f(x_a)} \leq \frac{5}{8}c$. Otherwise (there is no adjacent edges in G), the optimal solution is obvious: simply take edges as many as possible (the degree of each vertex is less or equal to one). □

4. Application to a small example and comparisons

In this section, we apply our algorithm to a small example, then we give for several values of c the worst-case ratios of our algorithm and compare them with the ones previously obtained. Let us apply Algorithm 3 to the instance of DSP described in Figure 3, where $n = 8$ and $k = 5$. By solving ($\underline{PL}$) (step 1) we get $x^L = \left(\frac{5}{8}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}\right)$ and $f_L\left(x^L\right) = 10$. Here we have $x^{L\beta} = x^L$ (step 2). Then (step 3, Algorithm 1), by considering successively the pairs $\{v_1, v_3\}$,$\{v_1, v_4\}$,$\{v_2, v_5\}$, $\{v_4, v_7\}$ and $\{v_5, v_6\}$ of non adjacent vertices, we get the feasible solutions of ($\underline{P}$): $\left(\frac{1}{4}, \frac{5}{8}, 1, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}\right)$, $\left(0, \frac{5}{8}, 1, \frac{7}{8}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}\right)$, $\left(0, 1, 1, \frac{7}{8}, \frac{1}{4}, \frac{5}{8}, \frac{5}{8}, \frac{5}{8}\right)$, $\left(0, 1, 1, 1, \frac{1}{4}, \frac{5}{8}, \frac{1}{2}, \frac{5}{8}\right)$, and finally $x^Q = \left(0, 1, 1, 1, 0, \frac{7}{8}, \frac{1}{2}, \frac{5}{8}\right)$ (the sets V_1 and V_2 are indicated in Figure 3).

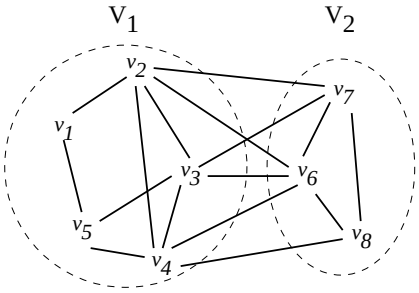

Figure 3. *Application of Algorithm 3: a small example.*

Finally (step 4), we get x^P from x^Q by Algorithm 2: here $C_6 = 3$, $C_7 = 2$, $C_8 = 1$, thus $x^P = (0, 1, 1, 1, 0, 1, 1, 0)$ and $f\left(x^P\right) = 9$. In this example, x^P is optimal (no integer solution can reach the value 10).

Now, we give in Table 1 our deterministic worst-case ratios when $n = ck$. "Naive" is the ratio obtained in Theorem 2.1. This ratio was improved in (Asahiro et al. 2000) by doing a tighter analysis of the greedy algorithm (column "Tighter"). "SDP Rand" is the expected ratio obtained by using the randomized algorithm based on semidefinite programming in (Jäger and Srivastav 2005). "PL Rand" is the expected ratio achieved by the randomized Goemans' algorithm (see Section

2). "PL" is our ratio which uses linear programming (see Theorem 3.5 and corollary 3.6).

	Naive	Tighter	SDP Rand	PL Rand	PL
Ratio	$\frac{n(n-1)}{k(k-1)} \geq c^2$	R		c	$\frac{f(x^*)}{f(x_a)} \leq \frac{9}{8}c$
c=2	≥ 4	$2.25 + O\left(\frac{1}{n}\right)$	1.606	2	2.25
c=3	≥ 9	$4 + O\left(\frac{n}{k^2}\right)$	2.227	3	3.375
c=4	≥ 16	$6 + O\left(\frac{n}{k^2}\right)$	2.943	4	4.5

TABLE 1. *Worst-case ratios for DSP*

As expected, using linear programming improves R which is obtained by using only a combinatorial algorithm (Theorem 2.1). The semidefinite approach provides a better (expected) ratio but involves a higher computational cost (moreover, it is a randomized algorithm). Let us point out that although our worst case ratio ($\frac{8}{9c}$) is slightly less than $\frac{1}{c}$, the complexity bottleneck of our deterministic approximation algorithm is that of solving the linear program ($\underline{PL}$) (see Section 3). This should be compared with the randomized approaches since a complete derandomization has a high computational cost and/or looses a significant part in the ratio $\frac{1}{c}$ for arbitrary k and n (see Section 2). For instance, the expected worst-case ratio of the partially derandomized algorithm of (Han et al. 2000) is better than our deterministic worst-case ratio only when k is larger than 4481 (one must have $\left(\frac{1}{1+k^{-\frac{1}{3}}}\right)^2 \geq \frac{8}{9}$) and for very large n.

5. Conclusion

We have proposed a simple deterministic approximation algorithm for the Densest-Subgraph Problem, which complexity is only the one of linear programming. From a practical point of view, our algorithm is very easy to implement and can provide good solutions for large instances of DSP. However, we conjecture that using such a linear programming approach one can not obtain a worst-case ratio that beats the target $\frac{1}{c}$. Recall that some random algorithms based on semidefinite programming have succeed in achieving better ratio (Feige et al. 2001b, Jäger and Srivastav 2005). But one has to solve a semidefinite program and to derandomize the algorithm to get the performance guarantee. Hence, a question remains: is there an efficient deterministic polynomial-time $\frac{1}{c}$-approximation algorithm for DSP?

References

Asahiro,Y. Iwama, K. Tamaki, H. and Tokuyama, T. 2000, 'Greedily finding a dense subgraph', *Journal of Algorithms*, vol. 34, no. 2, pp. 203-221.

Arora, S. Karger, D. and Karpinski, M. 1999, 'Polynomial time approximation schemes for dense instances of NP-hard problems', *Journal of Computer and System Sciences*, vol. 58, no. 1, pp. 193-210.

Carlson, R. and Nemhauser, G. 1966, 'Clustering to minimize interaction costs', *Operations Research*, vol. 14, pp. 52-58.

Corneil, D.G. and Perl, Y. 1984, 'Clustering and domination in perfect graphs', *Discrete Applied Mathematics*, vol. 9, no. 1, pp. 7-39.

Feige, U. Kortsarz, G. and Peleg, D. 2001, 'The dense k-Subgraph Problem', *Algorithmica*, vol. 29, no. 3, pp. 410-421.

Feige, U. Langberg, M. 2001, 'Approximation Algorithms for Maximization Problems Arising in Graph Partitioning', *Journal of Algorithms*, vol. 41, no. 2, pp. 174-211.

Feige, U. Langberg, M. 2001, 'The RPR^2 rounding technique for semidefinite programs', *ICALP'2001*, Crete, Greece, Lecture Notes in Computer Science No 2076, pp. 213-224, Springer, Berlin.

Freize, A. and Jerrum, M. 1997, 'Improved approximation algorithms for MAX k-CUT and MAX BISECTION', *Algorithmica*, vol. 18, pp. 67-81.

Goemans, X. and Williamson, D.P. 1995, 'Improved approximation algorithms for maximum-cut and satisfiability problems using semidefinite programming', *Journal of ACM*, vol. 42, no. 6, pp. 1115-1145.

Han, Q. Ye, Y. and Zhang, J. 2000, 'Approximation of Dense-k-Subgraph', Working Paper, Department of Management Sciences, Henry B. Tippie College of Business, The University of Iowa, Iowa City, IA 52242, USA.

Han, Q. Ye, Y. and Zhang, J. 2002, 'An improved rounded Method and Semidefinite Programming Relaxation for Graph Partition', *Mathematical Programming*, vol. 92, no. 3, pp. 509-535.

Mahajan, S. Hariharan, R. 1999, 'Derandomizing approximation algorithms based on semidefinite programming', *SIAM Journal on Computing*, vol. 28, no. 5, pp. 1641-1663.

Hassin, R. Rubinstein, S. and Tamir, A. 1997, 'Approximation algorithms for maximum dispersion', *Operations Research Letters*, vol. 21, pp. 133-137.

Jäger, G. and Srivastav, A. 2005, 'Improved approximation algorithms for maximum graph partitioning problems', *Journal of Combinatorial Optimization*, vol. 10, no. 2, pp. 133-167.

Kortsarz, G. and Peleg, D. 1993, 'On choosing a dense subgraph', *34th Annual IEEE Symposium on Foundations of Computer Science*, Palo Alto, California, pp. 692-701.

Krarup, J. Pisinger, D. and Plastria, F. 2002, 'Discrete location problems with push-pull objectives', *Discrete Applied Mathematics*, vol. 123, pp. 363-378.

Raghavan, P. Thompson, C. 1987, 'Randomized Rounding: a technique for provably good algorithms and algorithmic proofs. Probabilistic construction of deterministic algorithms', *Combinatorica*, vol. 7, pp. 365-374.

Raghavan, P. 1988, 'Probabilistic construction of deterministic algorithms. Approximate packing integer problems', *Journal of Computer and System Sciences*, vol. 37, no. 2, pp. 130-143.

Sahni, S. and Gonzalez, T. 1976, 'P-complete approximation problems', *Journal of ACM*, vol. 23, pp. 555-565.

Srivastav A. 2001, 'Derandomization in Combinatorial Optimization', In *Handbook of Randomized Computing*, Volume II, Chapter 18, pp. 731-842, Pardalos, Rajasekaran, Reif, Rolim (eds.), Kluwer Academic Publishers.

Srivastav, A. and Wolf, K. 1998, 'Finding Dense Subgraphs with Semidefinite Programming', *International Workshop on Approximation'98*, Lecture Notes in Computer Science vol. 1444, pp. 181-191.

An erratum: Srivastav, A. and Wolf, K. 1999, 'Erratum on Finding Dense Subgraphs with Semidefinite Programming', *Preprint, Mathematisches Seminar, Universitaet zu Kiel, 1999.*