Approximation Algorithms for the Generalized Assignment Problem

Prachi Shah

1 Introduction

In this project, we will be focusing on the Generalized Assignment Problem (GAP) where n jobs are to be assigned to m machines or agents. Each job j, incurs a cost c_{ij} and has processing time p_{ij} when run on any machine i. Additionally, each machine has a limit on the total operational time T_i. By appropriate scaling of processing times, we can assume without loss of generality that $T_i = T$ for all i. The goal is to assign all jobs to machines while minimizing the total cost and ensuring that the total processing time of jobs assigned to a machine does not exceed the time limit. GAP can be formulated as the following integer program,

$$\begin{aligned} \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} & & \\ \text{subject to} \quad & \sum_{i=1}^{m} x_{ij} = 1 & \text{for all } j \in \{1,\dots,n\} & \qquad \text{(GAP)} \\ & \sum_{j=1}^{n} p_{ij}x_{ij} \le T & \text{for all } i \in \{1,\dots,m\} & \\ & x_{ij} \in \{0,1\} \quad \text{for all } i,j \in \{1,\dots,m\} \times \{1,\dots,n\} & & \end{aligned}$$

Even the feasibility problem for GAP, that is the problem of determining if there exists a feasible solution to any given instance of GAP, is known to be NP-Hard. Shmoys and Tardos [1993] provide a polynomial time, bi-partite matching based linear programming rounding algorithm that gives a solution with objective value at most the optimal objective, but the operational time at any machine is at most $2T$. Lau et al. [2011] re-prove this approximation result using an iterative rounding algorithm inspired by Jain [2001].

An alternate version of this problem that is well studied is the maximization problem, with the relaxation that only a subset of jobs may be assigned to the machines. We refer to this problem as max-GAP and it can be formulated as follows,

$$\begin{aligned} \max \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} & & \\ \text{subject to} \quad & \sum_{i=1}^{m} x_{ij} \le 1 & \text{for all } j \in \{1,\dots,n\} & \qquad \text{(max-GAP)} \\ & \sum_{j=1}^{n} p_{ij}x_{ij} \le T & \text{for all } i \in \{1,\dots,m\} & \\ & x_{ij} \in \{0,1\} \quad \text{for all } i,j \in \{1,\dots,m\} \times \{1,\dots,n\} & & \end{aligned}$$

This problem is trivially feasible, however, it remains NP-Hard. Fleischer et al. [2006] provide a LP rounding algorithm with $(1 - 1/e)$ approximation bound and a local search algorithm with $1/2$ approximation guarantee. Feige and Vondrak [2006] show that the $(1 - 1/e)$ approximation bound is not tight and use randomization to achieve an approximation of $(1 - 1/e + \epsilon)$ for a fixed constant $\epsilon > 0$. Cohen et al. [2006] use approximation algorithms for knapsacks to develop an $(1 + \alpha)$ approximation algorithm for GAP where α is the approximation bound of the algorithm for the knapsack problem.

The goal of this project is to empirically compare the performance of approximation algorithms for max-GAP. We consider the following algorithms for this purpose -

1. Bipartite matching based LP Rounding due to Shmoys and Tardos [1993] ($\alpha = 1/2$)
2. Iterative LP Rounding due to Lau et al. [2011] with approximation bound ($\alpha = 1/2$)
3. Local Search due to Fleischer et al. [2006] with approximation bound ($\alpha = 1/2$)
4. Configuration LP Rounding due to Fleischer et al. [2006] with approximation bound ($\alpha = 1 - 1/e$)
5. Naive LP Rounding heuristic ($\alpha = 0$)

Here α refers to the approximation ratio for the algorithm. The rest of this report is organized as follows. In Section 2 we provide an overview of the algorithms and discuss how the LP Rounding algorithms for GAP are translated to the max-GAP problems. In Section 3 we provide computational details and discuss the results.

2 Algorithms for max-GAP

This section provides details regarding the algorithms considered in the project.

2.1 Bipartite matching based LP Rounding Algorithm

Let $p_i(J)$ and $c_i(J)$ denote the total processing time and total costs of all jobs in $J \subseteq [n]$ on machine i. We will now show how the bipartite matching based LP Rounding algorithm for GAP can be translated to a $1/2$ approximation algorithm for max-GAP problem. The modified algorithm is as follows -

1. Solve the LP relaxation, let x^* be the solution and $k_i = \lceil \sum_{j=1}^{n} x^*_{ij} \rceil$
2. Construct the bipartite graph using the process described in Shmoys and Tardos [1993]
3. Solve the linear program for the matching problem on this bipartite graph to obtain an integral optimal basic feasible solution
4. For each machine, let j_i be the unique job assigned to the first slot of machine i, and let J_i be the set of all other jobs assigned to machine i. Then,
   - If $p_{ij_i} + p_i(J_i) \leq T$: assign all jobs in $J_i \cup \{j_i\}$ to machine i
   - Else if $c_{ij_i} > c_i(J_i)$: assign only j_i to machine i
   - Else assign all jobs in J_i to machine i

Let $F_i \subseteq [n]$ to denote the set of jobs assigned to machine i be the algorithm. Note that while in the case of GAP, a complete matching problem was solved on the bipartite graph, for max-GAP

we solve relax it to the matching problem.

Lemma 1. *The bipartite matching based LP Rounding algorithm for the max-GAP problem return a feasible solution with objective value at least* 1/2 *OPT, where OPT is the optimal solution for the problem.*

Proof. Fix any machine i. From the proof of the approximation result in Shmoys and Tardos [1993], we know that $p_i(J_i) \leq T$ and $p_i(J_i) \leq T$ and therefore feasibility follows directly. Due to the criteria of selection between j_i and J_i,

$$c_i(F_i) = \max\{\ c_{ij_i},\ c(J_i)\ \} \geq\ \frac{1}{2}\ c_{ij_i} + \frac{1}{2}\ c(J_i)$$

Moreover, since the solution of the LP relaxation remains a feasible solution to the bipartite matching LP, we also know that,

$$\sum_{i=1}^{m} \left(c_{ij_i} + c(J_i)\right) \geq c_{LP}$$

where, c_{LP} is the objective value of the LP relaxation. We therefore have,

$$\sum_{i=1}^{m} c_i(F_i) \geq \sum_{i=1}^{m} \left(\frac{1}{2}\ c_{ij_i} + \frac{1}{2}\ c(J_i)\right) \geq \frac{1}{2}\ c_{LP} \geq \frac{1}{2}\ \text{OPT}$$

□

2.2 Iterative LP Rounding Algorithm

The translation for the iterative algorithm from GAP as presented in Lau et al. [2011] to max-GAP is similar to the translation of the bipartite matching based LP rounding. Consider the following slightly modified linear program,

$$\begin{aligned}
LP_{ga} = \ \max \ & \sum_{(i,j)\in E} c_{ij}x_{ij} & \\
\text{subject to} \quad & \sum_{e\in\delta(j)} x_{ij} \leq 1 & \text{for all } j \in J' \\
& \sum_{e\in\delta(i)} p_{ij}x_{ij} \leq T_i & \text{for all } i \in M' \\
& x_e \in \{0,1\} & \text{for all } e \in E
\end{aligned}$$

where $\delta(v)$ are all edges in E incident on v. We also denote $d_x(v) = \left|\{e \in E\ :\ u \in e,\ x_e > 0\}\right|$. The max-GAP algorithm is as follows -

1. Initialize $J_i \leftarrow \emptyset$, $O_i \leftarrow \emptyset$, $M' \leftarrow [m]$, $J' \leftarrow [n]$, $T_i \leftarrow T$, $E \leftarrow \{(i,j) : p_ij \leq T\}$
2. While $J' \neq \emptyset$,
   (a) Find an optimal extreme point to LP_{ga} and remove all edges (i,j) such that $x_{ij} = 0$.
   (b) If there is a variable such that $x_{ij} = 1$ and $i \in M'$, update $J_i \leftarrow J_i \cup \{j\}$, $T_i \leftarrow T_i - p_{ij}$, $J' \leftarrow J' \setminus \{j\}$.
   (c) If there is a variable such that $x_{ij} = 1$ and $i \notin M'$, update $O_i \leftarrow O_i \cup \{j\}$, $J' \leftarrow J' \setminus \{j\}$.

(d) If there is a machine i with $d_x(i) = 1$ or a machine with $d_x(i) = 2$ and $\sum_{e\in\delta(i)} x_e \geq 1$, update $M' \leftarrow M' \setminus \{i\}$

3. Assign jobs to machines as follows -

(i) If $O_i = \emptyset$, assign all jobs in J_i to machine i.

(ii) If $O_i = \{j_i\}$

- If $p_i(O_i) + p_i(J_i) \leq T$: assign all jobs in $J_i \cup \{j_i\}$ to machine i
- Else if $c_i(O_i) > c_i(J_i)$: assign only j_i to machine i
- Else assign all jobs in J_i to machine i

(iii) If $O_i = \{j_{i1}, j_{i2}\}$, without loss of generality assume $p_{ij_{i1}} \geq p_{ij_{i2}}$. Update $J_i \leftarrow J_i \cup \{j_{i2}\}$, $O_i \leftarrow O_i \setminus \{j_{i2}\}$ and apply the same rules as the previous case.

Lemma 2. *The Iterative LP Rounding algorithm for the max-GAP problem return a feasible solution with objective value at least* $1/2$ *OPT, where OPT is the optimal solution for the problem.*

Proof. We first claim that all results in the proof for GAP applies to the algorithm for max-GAP. The only part of the proof that requires $\sum_{i\in M'} x_{ij} = 1$ is to show that if (a) to (c) in step 2 are not true, condition (d) must hold. However, as an intermediate result, the proof shows that if (a) to (c) don't hold, all constraints in the LP are tight due to the rank lemma. The rank lemma also apply to the max-GAP problem and therefore all results in the proof of GAP apply to max-GAP.

Let $F_i \subseteq [n]$ to denote the set of jobs assigned to machine i be the algorithm. The algorithm for GAP assigns all jobs in $J_i \cup O_i$ to machine i. From the proof for the same, and using the notation defined previously, we know that,

$$\sum_{i=1}^{m} c(J_i) + c(O_i) \geq c_{LP} \geq \text{OPT}$$

To prove the lemma, it is sufficient to show that for all machines,

$$p_i(F_i) \leq T \text{ and } c_i(F_i) \geq (c(J_i) + c(O_i))/2.$$

We will now analyse the cases (i) to (iii) individually. First, observe that since J_i is updated only as long as the constraint for machine i is not relaxed, $p_i(J_i) \leq T$ for all i. When $O_i = \emptyset$, $F_i = J_i$, thus both feasibility and the objective bound trivially hold. When $O_i = \{j_i\}$, we have $p_i j_{i1} \leq T$, and therefore feasibility is ensured for all subcases. Moreover, since $c(F_i) = \max\{c_i(O_i), c_i(J_i)\}$, the required condition on the objective also holds. To prove these conditions for case (iii), we will show that $p(J_i \cup j_{i2}) \leq T$. Given this, following the same arguments as case (ii) proves the required result.

Note that given $|O_i| = 2$, implies that the constraint for machine i was relaxed. Consider the iteration in which it was relaxed. Then, when $d_x(i) = 2$ and therefore $x_{ij_1} + x_{ij_2} \geq 1$. We then have that,

$$\begin{aligned} p_i(J_i) + p_{ij_2} &\leq p_i(J_i) + p_{ij_2}(x_{ij_1} + x_{ij_2}) \\ &\leq p_i(J_i) + p_{ij_1}x_{ij_1} + p_{ij_2}x_{ij_2} \\ &\leq p_i(J_i) + T_i = T \end{aligned}$$

where the first inequality follows due to $x_{ij_1} + x_{ij_2} \geq 1$ and the second inequality follows since $p_{ij_2} \leq p_{ij_1}$. The last inequality follows due to the time limit constraint for i and the last equality due to the mechanism of updating T_i. □

2.3 Local Search Algorithm

We implement the local search algorithm as described in Fleischer et al. [2006]. The algorithm requires solving the knapsack problem as a subroutine. For the local search to be a truly polynomial time approximation scheme, it requires the knapsack problem to be solved in polynomial time $\beta-$approximate algorithm. However, for this project we solve it optimally using a pseudo-polynomial dynamic programming approach and thus $\beta = 1$.

Let F_i denoting all jobs assigned to machine i be an intermediate solution at any iteration. For each machine i, a locally improving solution is computed as follows,

1. For each item j, the marginal value of j is

$$w_{ij} = \begin{cases} c_{ij} - c_{i'j} & \text{if } j \in F_i \text{ for some } i' \neq i \\ c_{ij} & \text{otherwise.} \end{cases}$$

2. Then the locally improving solution S_i is the optimal solution to the knapsack instance

$$\max\{ \textstyle\sum_{j \in S} w_{ij} \mid \sum_{j \in S} p_{ij} \leq T,\ S \subseteq [n]\}$$

With local improvements defined above and a fixed $\epsilon > 0$ the local search algorithm for max-GAP is as follows,

- Initialize $\hat{x} \leftarrow 0$, $F_i \leftarrow \emptyset$
- For $(n/\beta)\ln(1/\epsilon)$ iterations, do
  - (a) Compute a locally improving solution S_i for every machine
  - (b) Let $\Delta_i = c_i(S_i) - c_i(F_i)$ for all machines
  - (c) Let $i^* = \arg\max_{i \in [m]} \Delta_i$
  - (d) If $\Delta_{i^*} > 0$, update $F_{i^*} \leftarrow S_{i^*}$ and for all $i \neq i^*$ update $F_i \leftarrow F_i \setminus S_{i^*}$

Fleischer et al. [2006] show that this algorithm is a $(\frac{\beta}{1+\beta} - \epsilon$ approximation algorithm for max-GAP. In our computational experiments, we take $\epsilon = 10^{-3}$

2.4 Configuration LP Rounding

Unlike the previous algorithms which were deterministic, the Configuration LP Rounding algorithm is randomized. We implement algorithm as described in Fleischer et al. [2006]. As in case of local search, this algorithm also requires solving the knapsack problem as a subroutine and we solve it similarly.

Let $\mathcal{S}_i = \{ S \subseteq [n] \mid \sum_{j \in S} p_{ij} \leq T\}$ be the set of all valid assignments on machine i. The

configuration LP for max-GAP is then formulated as follows,

$$\begin{aligned}
\max \quad & \sum_{i=1}^{m} \sum_{S \in \mathcal{S}_i} c_i(S)\, y_S^i \\
\text{subject to} \quad & \sum_{i \in [m]} \sum_{S \in \mathcal{S}_i : j \in S} y_S^i \leq 1 && \text{for all } j \in \{1, \ldots, n\} \\
& \sum_{S \in \mathcal{S}_i} y_S^i = 1 && \text{for all } i \in \{1, \ldots, m\} \\
& y_S^i \in \{0, 1\} && \text{for all } i \in \{1, \ldots, m\}, S \in \mathcal{S}_i
\end{aligned}$$

While this LP may have an exponential number of variables, the separation problem for its dual is the knapsack problem and can therefore be solved efficiently using ellipsoid algorithm. Equivalently, we solve it via column generation on the primal problem. The random rounding algorithm can then be summarized as follows,

1. Solve the configuration LP and let y_S^i be the solution.
2. For each machine i, assign $F_i \leftarrow S$ with probability y_S^i.
3. If some job j is assigned to multiple machines, pick the machine which maximizes $c_i j$ and remove j from all others.

Fleischer et al. [2006] show that in expectation, the value of the returned solution is at least $(1 - 1/e)\text{OPT}$.

2.5 Naive LP Rounding

We also compare the performance of the approximation algorithms against the following naive heuristic,

1. Solve the max-GAP linear program and let x be the solution
2. For each machine i, assign $F_i \leftarrow \{j \in [n] \,:\, x_{ij} = 1\}$

3 Computational Results

We test the algorithms on a set of 60 instances from the OR-Library Beasley [1990]. For each run of an instance with an algorithm, let $\hat{z}$ be the objective value of the solution return. We compute $\alpha = \hat{z}/\text{OPT}$ and compare the algorithms based on the distribution of α across instances in Fig. 1. The grey dashed lines in the plot represent the approximation ratios for the algorithms.

The performance of the Shmoys and Tardos's LP rounding algorithm as well as the Iterative LP rounding is better than Naive LP rounding. This should be expected since this algorithms implicitly consider the Naive LP rounding solution. The randomized rounding of the configuration LP solution is better than approaches using the compact LP formulation. This is again in line with expectations since the configuration LP has a tighter LP bound as well as the better approximation guarantee of the randomized rounding algorithm. The most noteworthy observation is that local search significantly out-performs all other algorithms.

Moreover, all algorithms performed much better than their approximation ratios. This may be largely due to the distributions of instances in consideration. The production times as well as

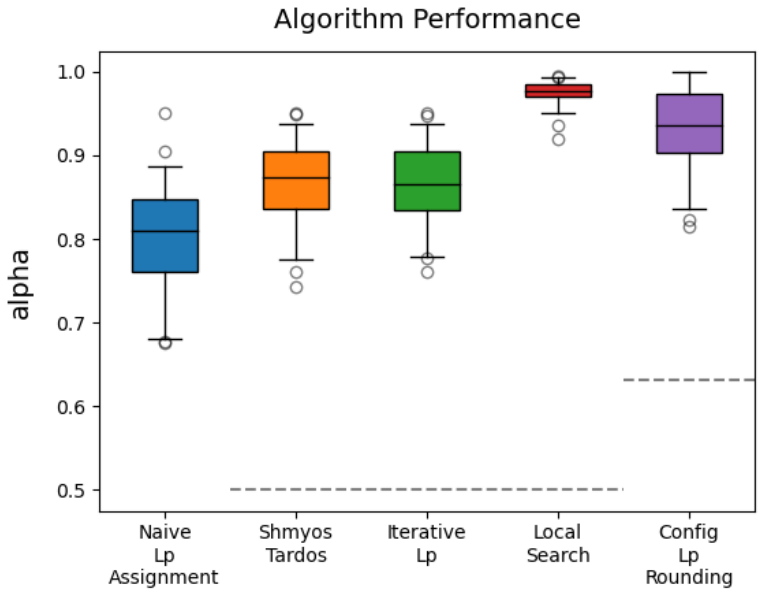

Figure 1: Comparison of algorithms for max-GAP on instances from OR-Library.

costs for these instances took comparable values across all instances and jobs. Given this fact, any maximal feasible assignment wouldn't have a very bad objective with a high probability. This may also explain the very good performance of the naive heuristic which has a median α value of more than 0.8.

References

John E. Beasley. Or-library: distributing test problems by electronic mail. *Journal of the operational research society*, 41(11):1069–1072, 1990.

Reuven Cohen, Liran Katzir, and Danny Raz. An efficient approximation for the generalized assignment problem. *Information Processing Letters*, 100(4):162–166, 2006.

Uriel Feige and Jan Vondrak. Approximation algorithms for allocation problems: Improving the factor of 1-1/e. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 667–676. IEEE, 2006.

Lisa Fleischer, Michel X Goemans, Vahab S Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *SODA*, volume 6, pages 611–620. Citeseer, 2006.

Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21:39–60, 2001.

Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.

David B Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical programming*, 62(1-3):461–474, 1993.